**Extending Cloud for Utilities (C4U CO)**

This document provides a step-by-step process to extend C4U CO while also providing in depth technical information that will allow you to extend C4U CO functionality for your use cases.

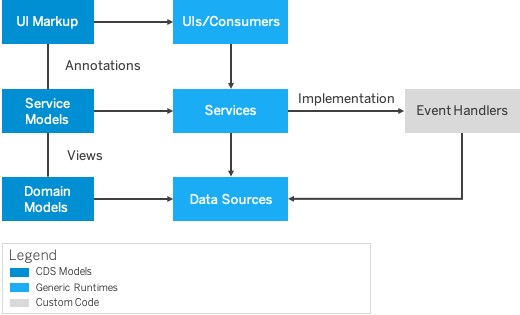
C4U CO is built upon the [SAP Cloud Application Programming Model framework](https://cap.cloud.sap/docs/) which allows C4U CO to focus on exposing its vast domain expertise by:

* Focusing on the core domain and domain logic.
* Basing design on the models of the domain.
* Evolving the models by always collaborating with domain experts.

**CDS**

Since the primary focus of the framework is through [domain driven design](https://martinfowler.com/bliki/DomainDrivenDesign.html) it makes sense to familiarize yourself with SAP’s [CDS](https://cap.cloud.sap/docs/cds/) universal modeling language. Once you are familiar with CDS it will become easier to create both service and database extensions.

CDS models are plain JSON objects that comply to the [Core Schema Notation](https://cap.cloud.sap/docs/cds/csn) (An open specification derived from JSON schema), which are in a very easily readable human format. The models are what drive the CAP service runtimes and databases as illustrated below:



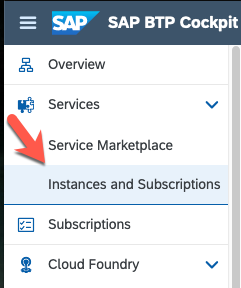
For complete documentation of CDS please refer to the following [link](https://cap.cloud.sap/docs/cds/cdl#entity-and-type-definitions).

**Subscribe to SAP Cloud for Utilities Foundation Retailer**

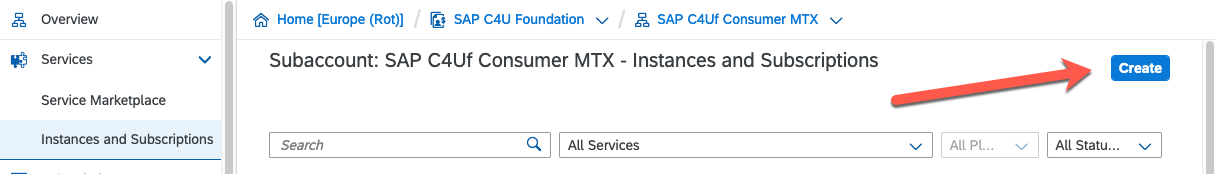
1. Go to <https://account.hana.ondemand.com> and login to your SAP BTP Subaccount from where you will be subscribing to SAP Cloud for Utilities Foundation Retailer.

Note: If you have more than one subaccount choose the one from where you wish to subscribe C4U

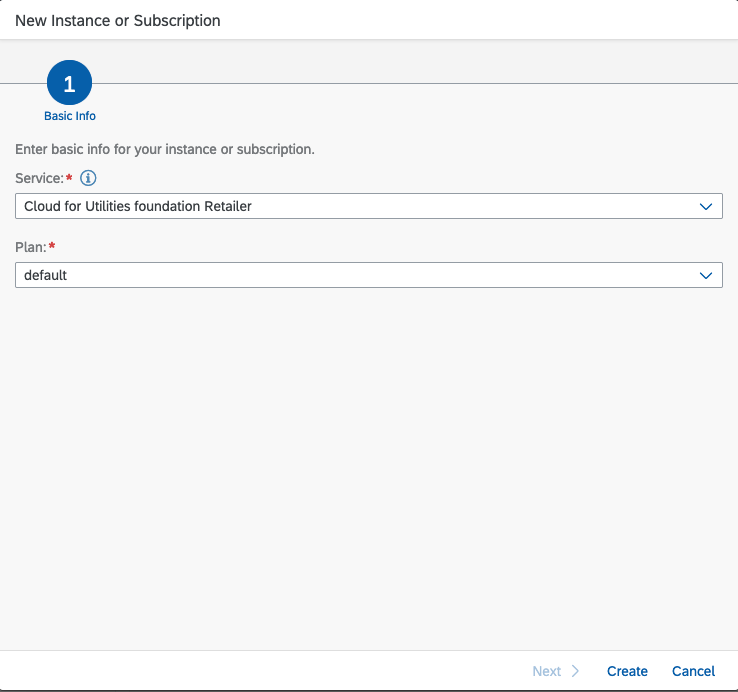
1. Choose “Instances and Subscriptions”



1. Subscribe to C4U from the SAP BTP Cockpit by clicking on “Create” button underneath your profile name.



1. A dialog box will appear asking which service you wish to subscribe to.
   1. Choose “Cloud for Utilities Foundation Retailer” from the dropdown.
   2. Select “default” as the plan.



1. After having subscribed to SAP Cloud for Utilities Foundation Retailer Development you will see

that the subscription is processing.



1. Once subscription has completed, the status will change from processing to completed.



**Service Broker Instances**

The service which implements the service broker API is referred to as the “Service Broker”. A service instance represents varies. In our context different service broker instances are created to offer different scopes required by the calling application.

For instance, you may wish to setup a service broker with only API Read or API Write rights. Extensibility is a special case and requires the following scopes depending on your situation:

1. ExtendCDS

Allows the extension developer to create and activate an extension.

1. ExtendCDSDelete

Allows the extension developer to delete tables created by a previous extension.

These scopes are not given by default by edom retailer and require a new service broker instance to be created with the above scopes to retrieve the proper jwt token for acess.

Before starting ensure that you have more than one entitlement.

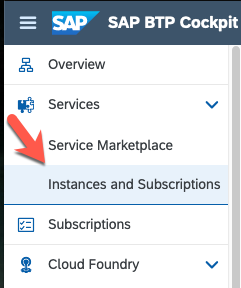
**Note:**

The same process can be used to create different credentials with different scopes. E.g., You may wish to create a service instance with API Read Only, or a combination of different scopes depending on how you wish to give access.

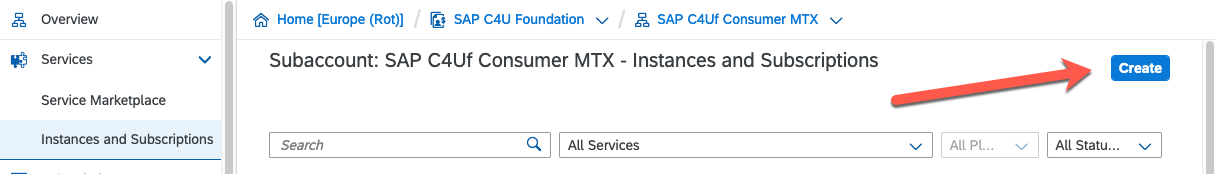
**Create an Instance of Service Broker for API Access**

In order to be able to access the API, we will create an instance of the Edom Retailer API that will give us the ability to query the released API.

1. From your subaccount choose “Instances and Subscriptions”



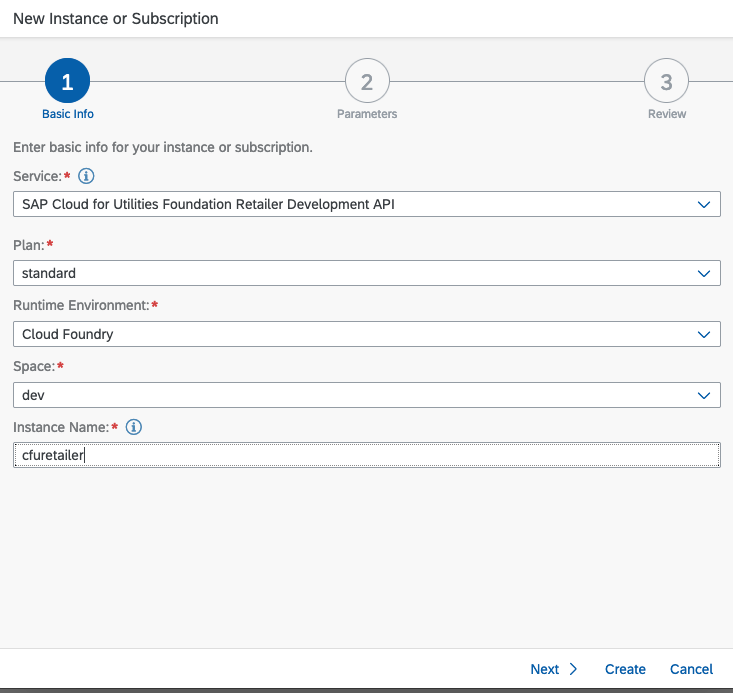
1. Create an instance by clicking on the Create button.



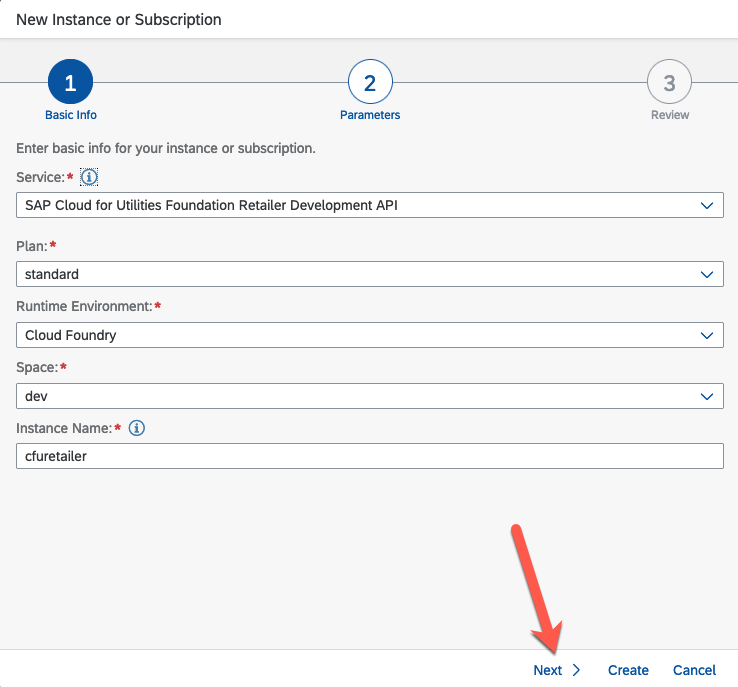
1. The following dialog will appear, choose the C4U CO Retailer API.

It will prompt to create a space in Cloud Foundry if you have not already done so.

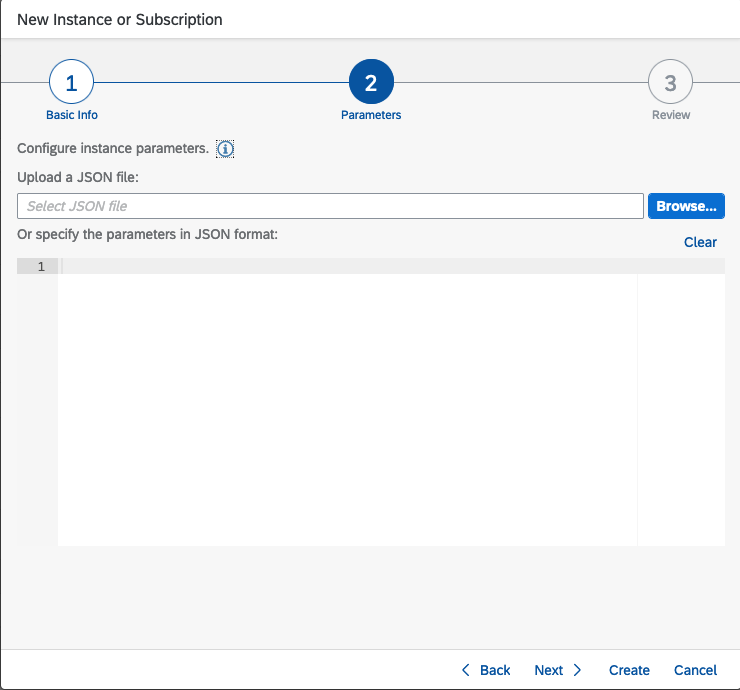
Give the instance a name. The instance name must be unique across our organization.



1. Since we do not want to use the default rights, hit Next to create our modified instance for extensibility.



1. The following screen dialog will appear:



1. This is the important step where we decide what scopes we will give to the instance. In this step we must provide json with the correct additional rights.

The format of the json is the following.

{

"xs-security": {

"xsappname": "c4u-foundation-retailer-dev-service-random-numbers",

"oauth2-configuration": {

"grant-types": [

"client\_x509",

"client\_credentials"

]

},

"authorities": [

Scopes you wish to add

]

}

}

The “xsappname” refers to the edom retailer application. Above this refers to the “dev-service” however in production it will be simply “service” therefore the xsappname would be "c4u-foundation-retailer-dev-service-<random-numbers>"

<Random numbers> is required to differentiate the service broker instance from others. You may wish even to set it up uniquely by giving it a unique suffix. (TODO: determine a naming convention).

Scopes you wish to add must be given in the following format:

"authorities" : [

"$XSMASTERAPPNAME.ExtendCDS",

"$XSMASTERAPPNAME.API.Read",

"$XSMASTERAPPNAME.API.Write"

]}

\*\* If this step is skipped the ExtendCDS scope will not be given to the jwt token. The developer will be able to still create extensions but will not be able to double check or view them. It is best add extra scopes for the extension developer in this case so they may be able to view and edit without having to retrieve a different token for other operations.

Therefore, create a json similar to:

{

"xs-security": {

"xsappname": "c4u-cn-edom-retailer-clone-789",

"oauth2-configuration": {

"grant-types": [

"client\_x509",

"client\_credentials"

]

},

"authorities": [

"$XSMASTERAPPNAME.API.Write",

"$XSMASTERAPPNAME.API.Read",

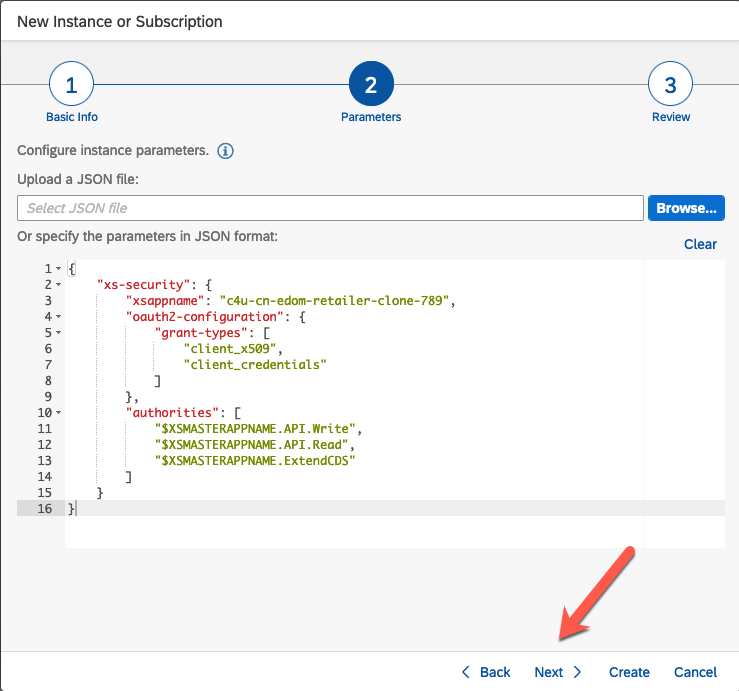
"$XSMASTERAPPNAME.ExtendCDS"

]

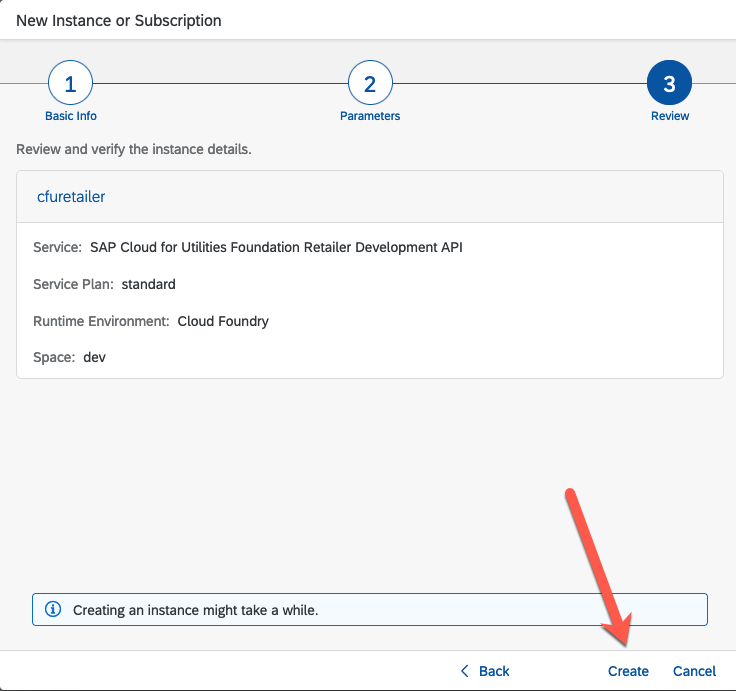
}

}

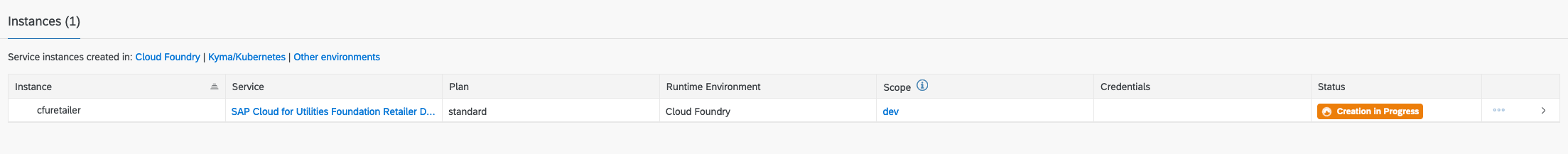
1. Add the Json to the dialog box:



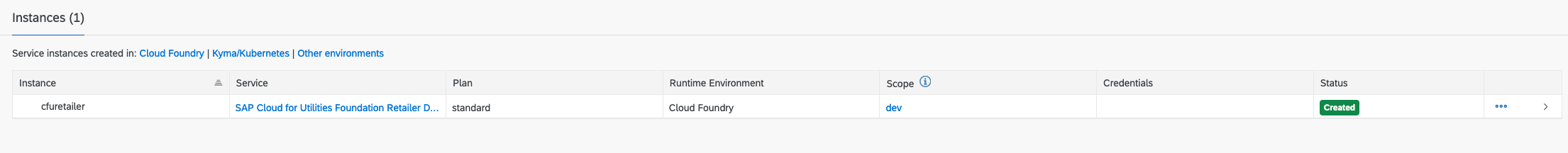
1. Double check your setting and then click Create



1. The following dialog will appear:

****

1. Our instance should now be created:

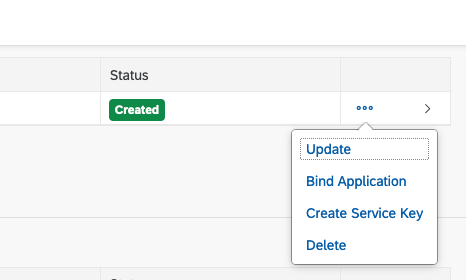


**Setup Service Key**

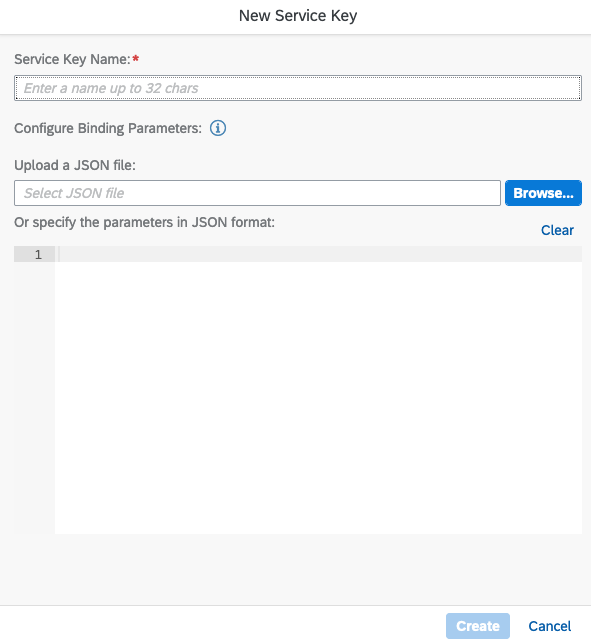
Now that we have created the instance we need to create a service key. With the service key we will be able to create an oauth token that will give us the ability to query our api.

1. From the Instances table, locate the row with our newly created instance name and left click the ellipsis (…).

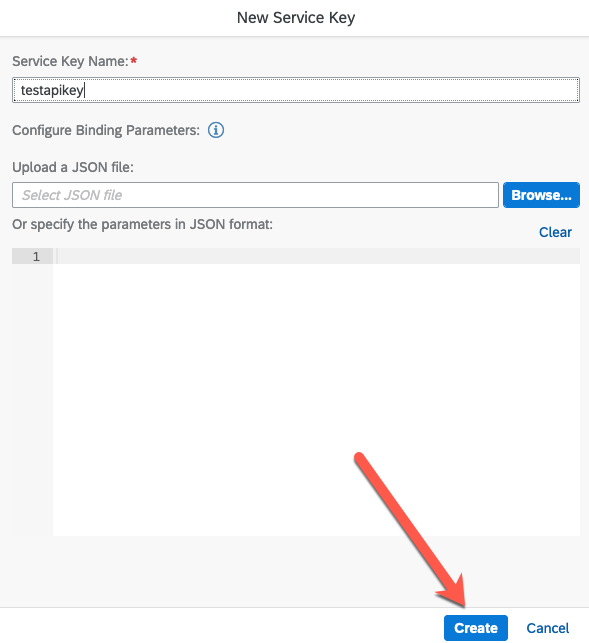
The following menu will appear:



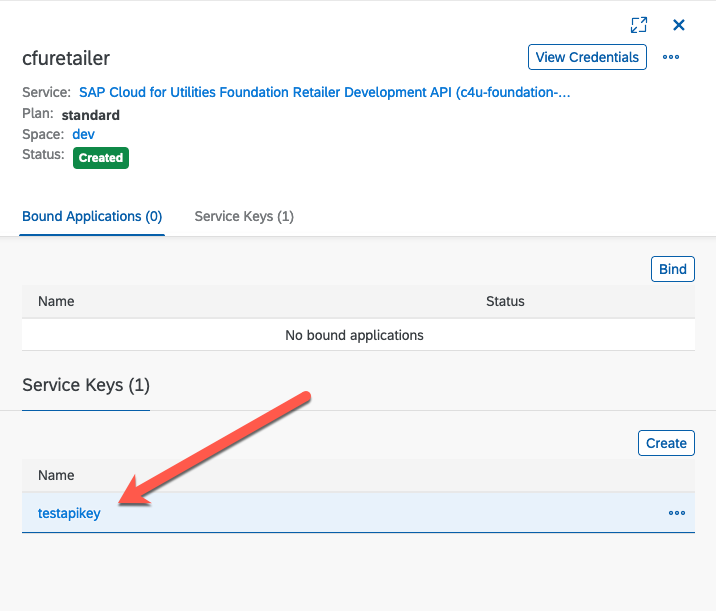
1. Choose “Create Service Key” and the following dialog will appear:



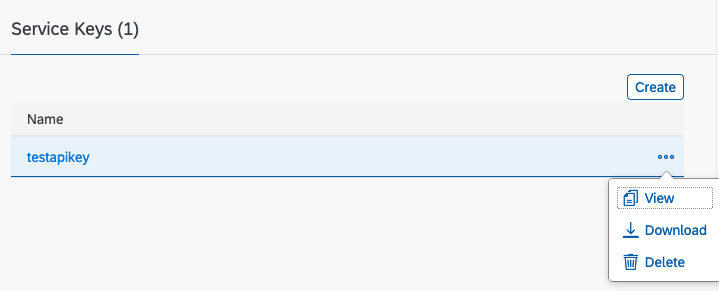
1. Choose a name for our service key and click “Create”



1. Our instance should now have a service key attached to it.



1. Left click on the ellipsis to view our service key credentials that have been created



1. Select “View” to see our credentials. The sections we are interested in are:

clientid, clientsecret and url.



You could also switch to “Form” View which might be easier to copy the required information:



**Create our Access Token**

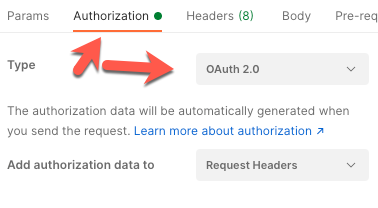
Now that we have successfully created our service key, we can use the above values to generate our Oauth2 access token. For this example, we will be using Postman and curl.

BTP reference documentation can be found here:

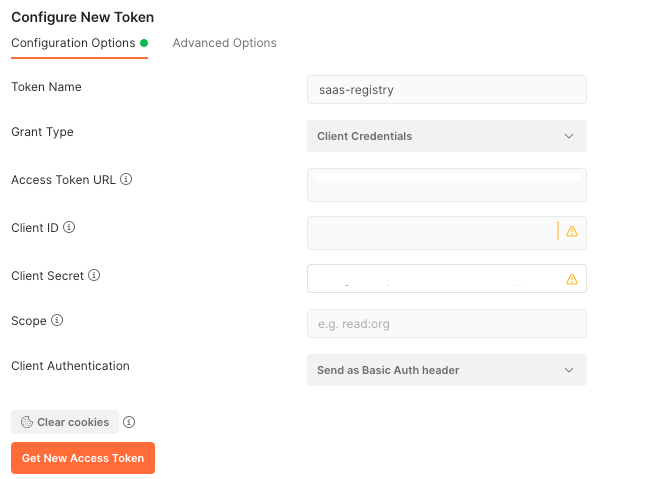
<https://help.sap.com/viewer/65de2977205c403bbc107264b8eccf4b/Cloud/en-US/6391b5dfe4704c6c8b71a32126828e9c.html>

Postman <https://www.postman.com/downloads/>

1. Start a new query and select the authorization tab. Choose as the type of authorization “Oauth 2.0”. For curl skip to the last step.



1. In the configure new token section add the information we have previously retrieved from the service key. Give your token a name and proceed filling in the required information.



1. Grant type = “Client Credentials”
2. Access token Url = (service key url) /oauth/token

E.g. if our service key url was :

<https://oursubaccount.authentication.eu10.hana.ondemand.com>

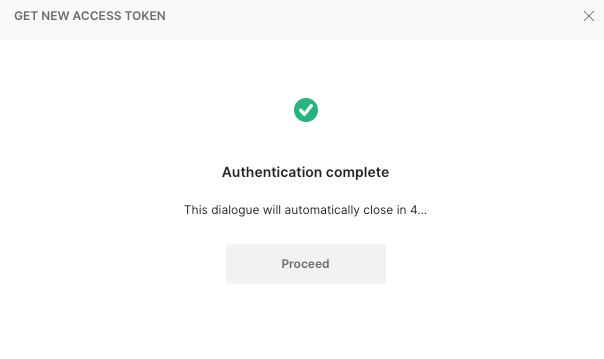
Then the access token url would be:

<https://oursubaccount.authentication.eu10.hana.ondemand.com/oauth/token>

1. Client ID = clientid from the service key
2. Cleitn Secret = clientsecret from the service key.
3. Click the “Get New Access Token” button.

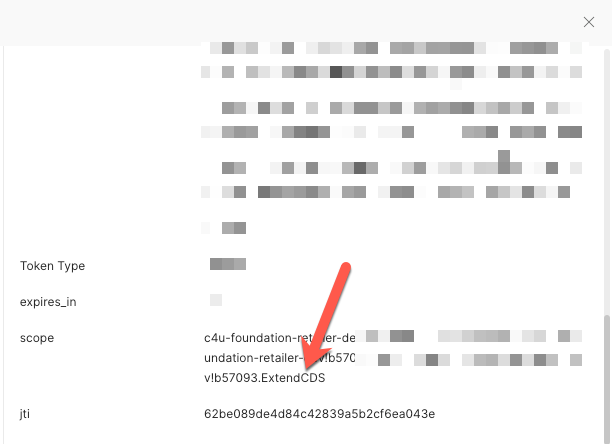
****

1. You should receive a valid token dialog:



1. Double check that you have the correct scopes for this token.

Scroll down in the Token Details dialog until you reach the scope key and verify that the “ExtendCDS” scope is there.



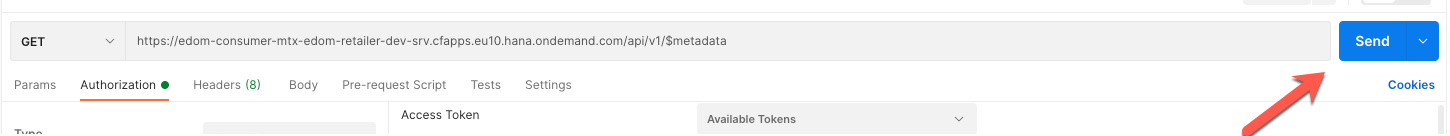
If the scope is not there, double check to make sure you created the service instance with the correct json payload. (See above)

1. Click the “Use Token”. You can now share this token with other requests based on the name you have given it.



1. Test a simple get.

Using your subdomain, you should be able to query for metadata to make sure our token is functioning properly.



1. In the response you should now see metadata returned from the Edom retailer.



1. If instead you need to retrieve the token using curl. Take the information you obtained previously for the clientid, clientsecret, url and plug them into the following command line:

curl --location --request POST '<url>/oauth/token' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--header 'Authorization: Basic <base64.encoded(client\_id:client\_secret)>' \

--data-urlencode 'client\_id=<client\_id>' \

--data-urlencode 'grant\_type=client\_credentials'

This will return the access token which you can use in another or request or plug into postman as the value for the

Authorization key.

Eg.

Bearer: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV\_adQssw5c

**Setup the Roles for your extension Developers.**

C4U CO comes with different roles that need to be assigned to extension developers. Currently there are two possible roles:

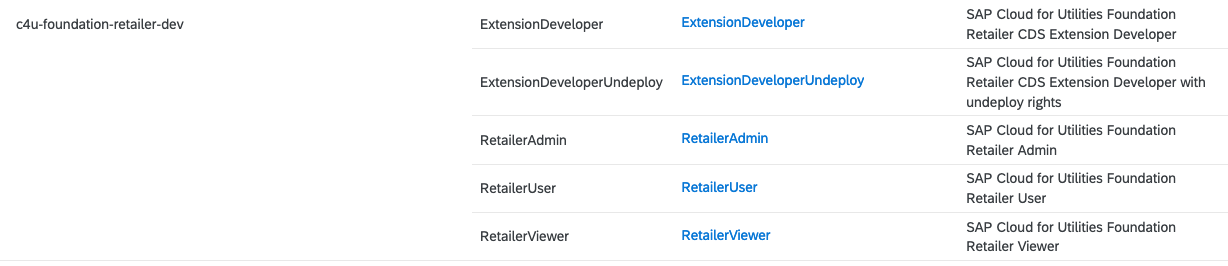
1. ExtensionDeveloper:

This role is intended for an extension developer to be able to develop an extension.

1. ExtensionDeveloperUndeploy:

This role is intended for an extension developer to be able to develop and extension and also be able to remove it.

As shown below:

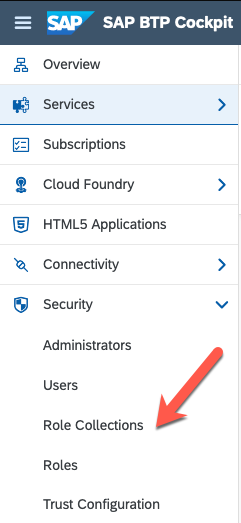


Note: It is also a best practice to assign an application role to the developer so they have the ability to run the application and view the extension in action.

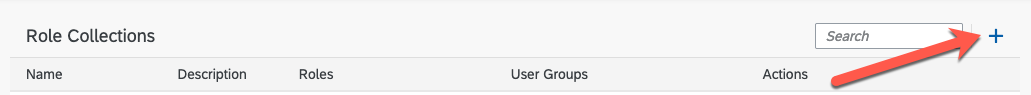
1. Create the Role Collection

Depending on the level of granulation you can decide to add other roles that are felt the developer should have, however for the purpose of developing extensions we will adhere only to the extension roles.

From BTP Cockpit choose Role Collections as shown below:



1. Create a new role collection by clicking the “+” button.



1. Give a name to the Extension Role Collection you are creating:

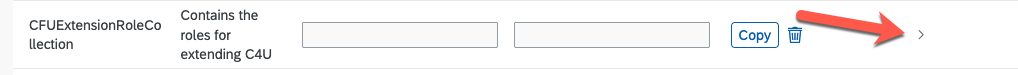


Click Create when you are finished.

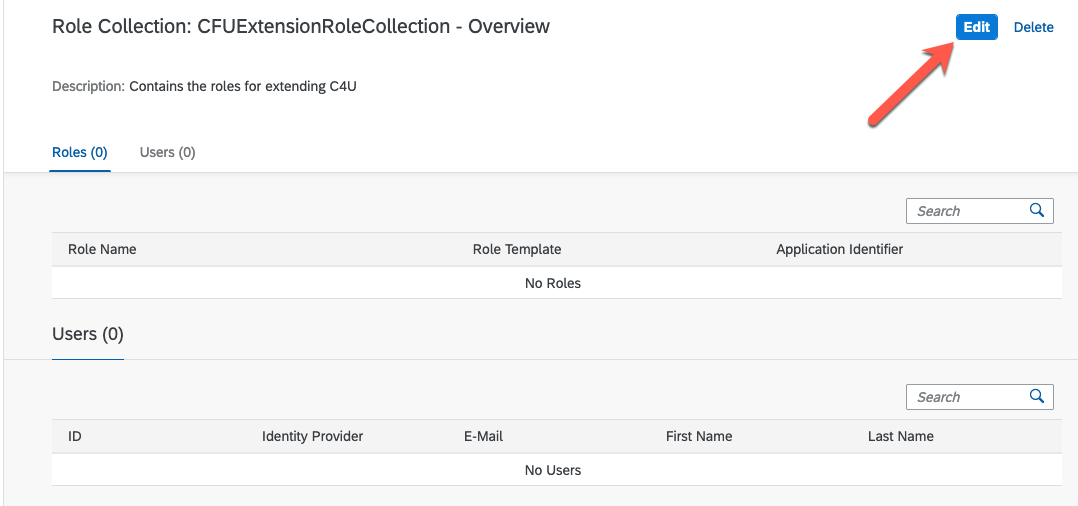
1. You will now see the newly created role collection in the role collection table.



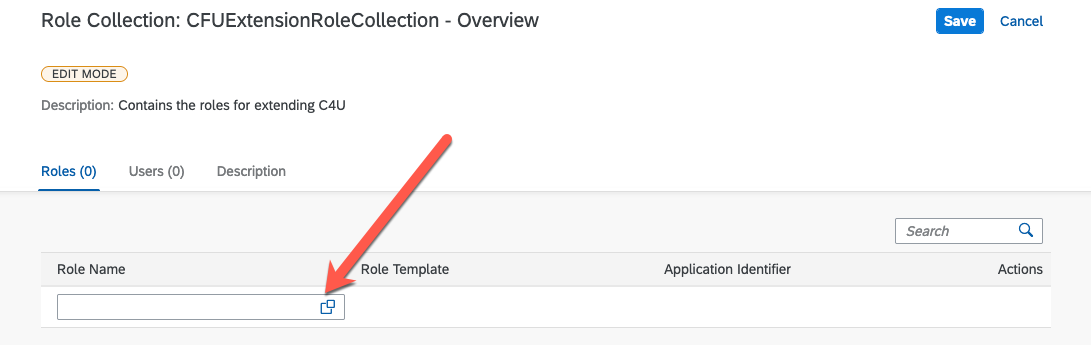
1. Click the Arrow to start adding the roles.



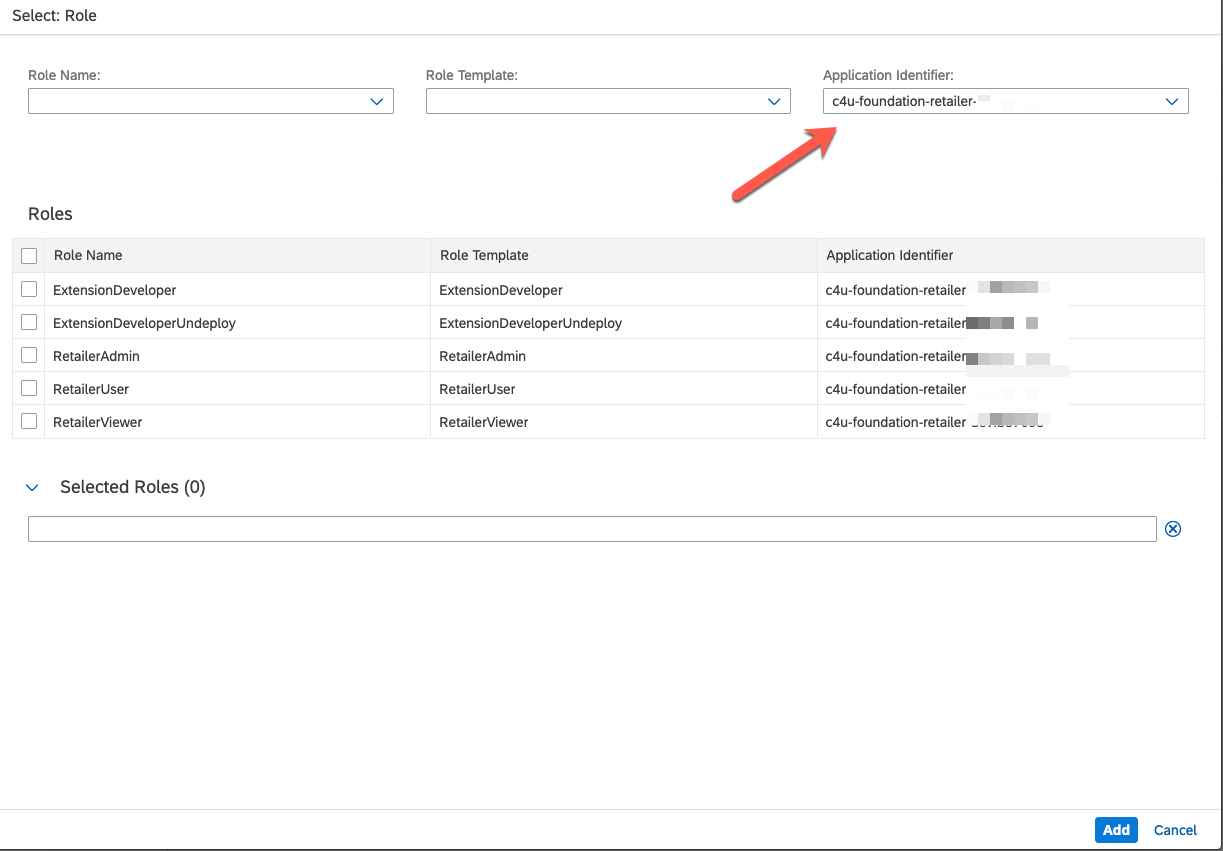
1. From the Role Collection Screen, click the Create button.



1. Choose the appropriate role names from the selection box.

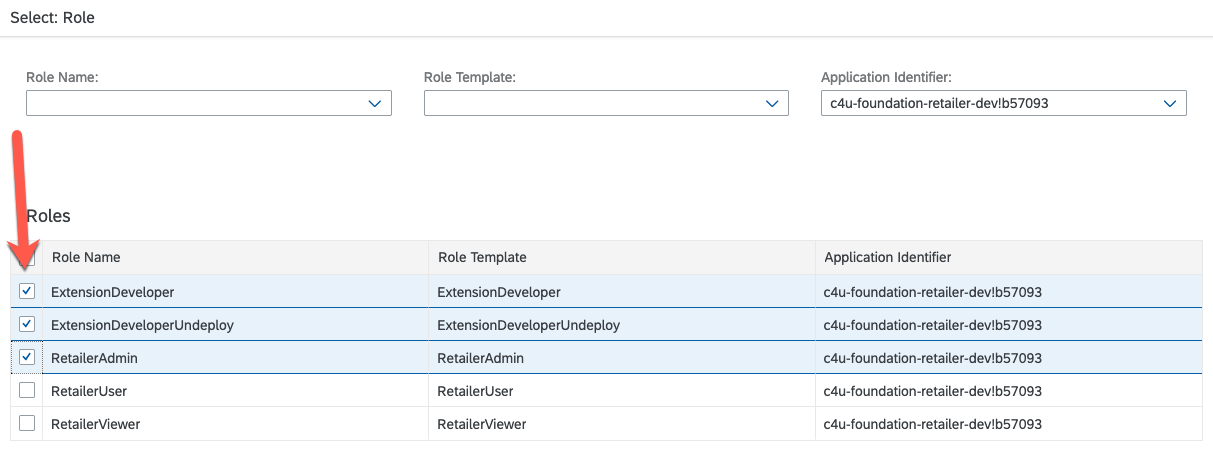


1. Filter your possible selections by choosing c4u-foundation-retailer from the Application Identifier.

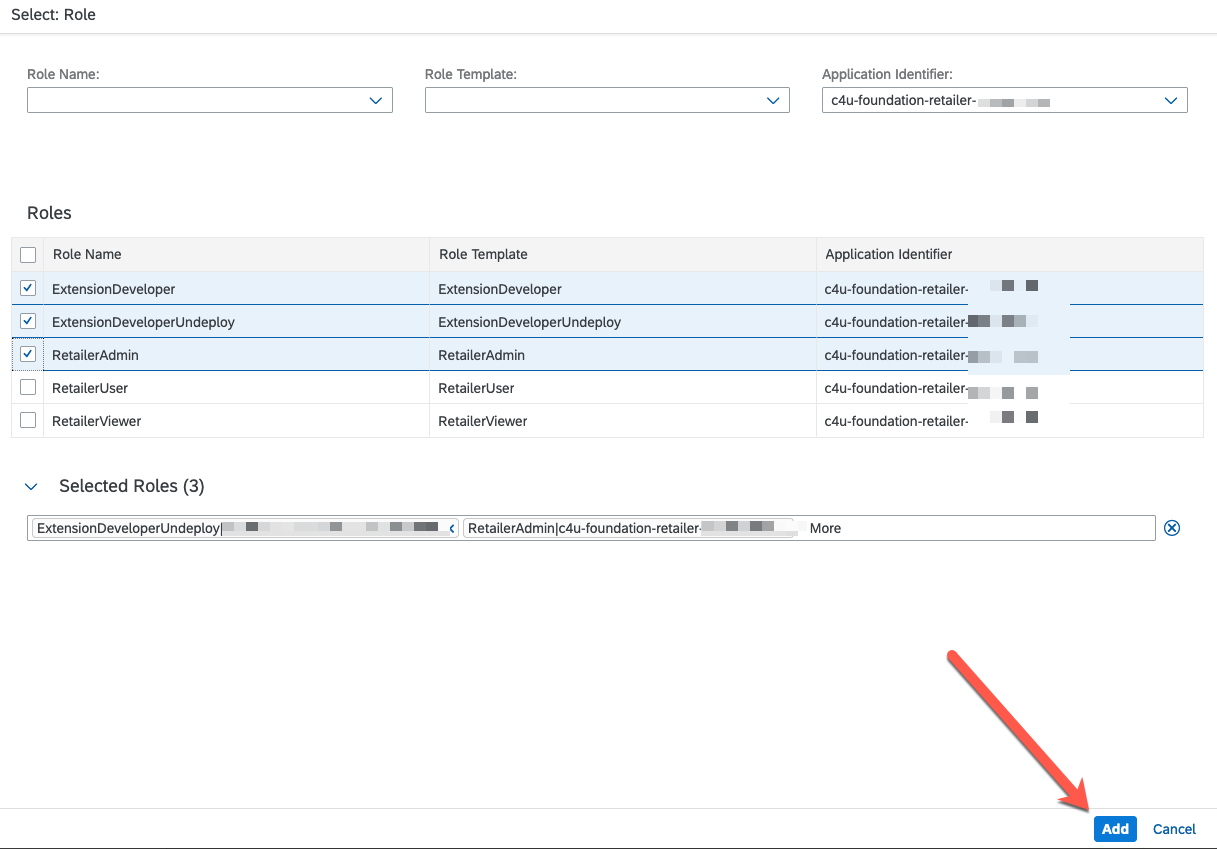


1. Select the roles for the Extension developer. Choose at a minimum the ExtensionDeveloper to develop extensions. If you wish the developer to be able to undeploy the extension then also choose the ExtensionDeveloperUndeploy Role.

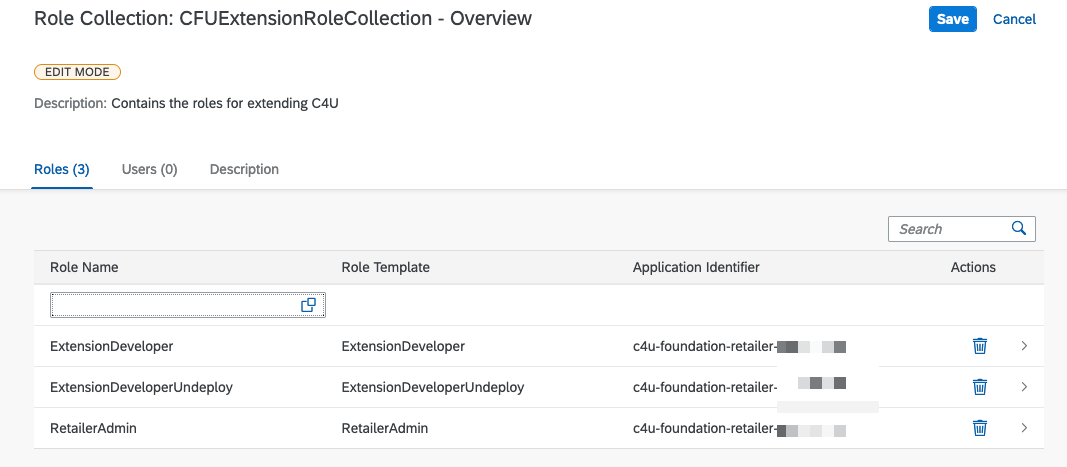
In our example we will also choose the RetailerAdmin, however this is only for demonstration purposes.



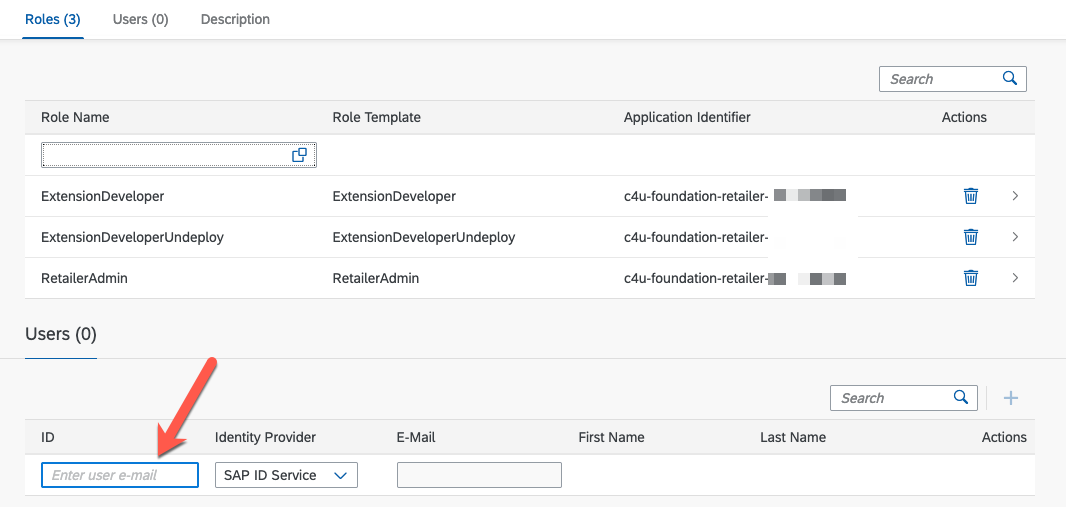
1. Select Add when you are finished.



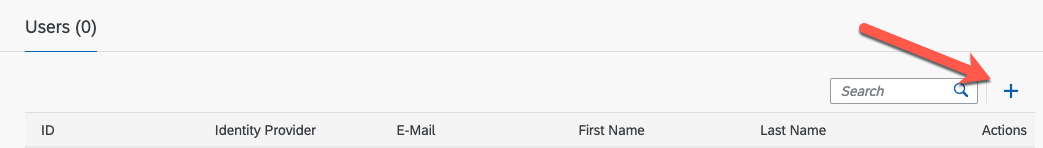
1. Your screen should now be similar to this:



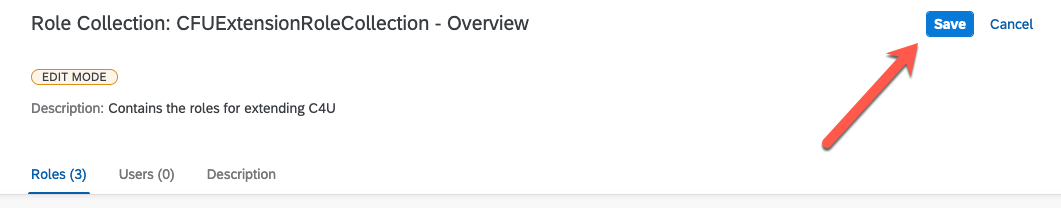
1. Now we can assign the new users to this role collection. If users have already been created in your subaccount goto the Assign Users section below.
2. Assign the users by entering in their emails



1. Select the users you wish to assign the roles to by Selecting “Users” underneath Security in the left-navigation menu and then click the “+” button

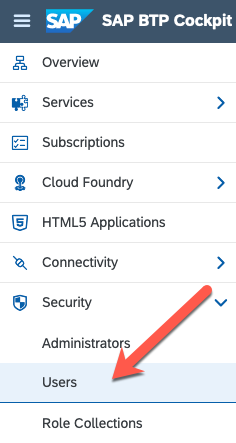


1. Click save to finish

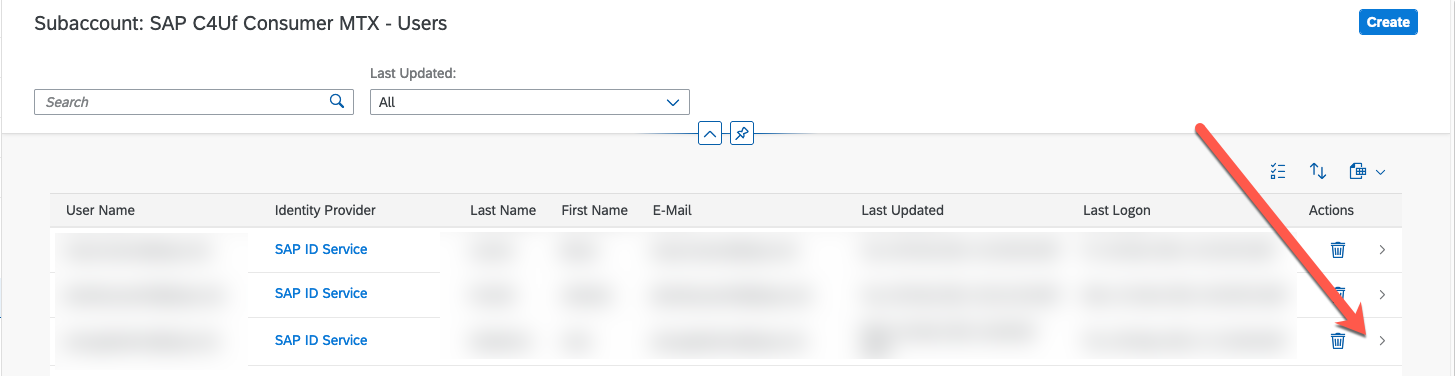


**Assign Existing Users**

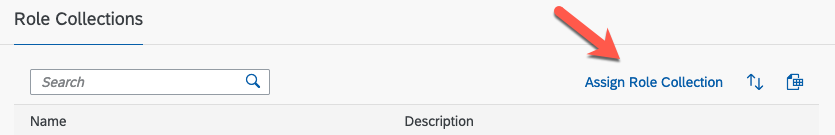
1. From the BTP Cockpit choose Users underneath the Security menu.



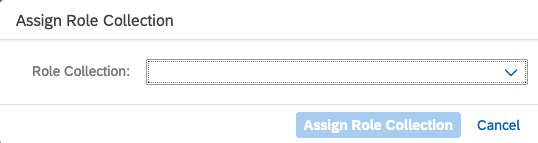
1. Select the user to assign the roles to from the user table by selecting the arrow “>” on the far right.



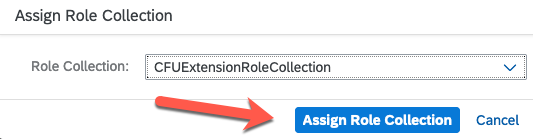
1. Select “Assign Role Collection”.



1. From the dialog box select the Extension roles you wish to assign to the user.



1. Use the previously created role collection to assign to your users by clicking the Assign Role Collection Button.



1. The developer(s) are now ready to start their extension.

These roles are defined as “ExtendCDS” and “ExtendCDSDelete”.

They are specified in the xsuaa settings defined in the mta.yaml of the application.

Once these have been added the developer can now start the extension.

**Project Setup**

Before starting the extension, project be sure to have the following setup

1. Either the SAP Business Application Studio or Visual Studio Code. You can find information on how to setup [here](https://cap.cloud.sap/docs/get-started/).
2. The cloud foundry command line [here](https://github.com/cloudfoundry/cli).

**Creating the Extension Project (VS Code) – Part I**

The extension is always derived from the default provider application (in this case, C4U CO Edom Retailer).

**Prerequisites:**

Before starting the extension project, there are a few prerequisites required:

Variables:

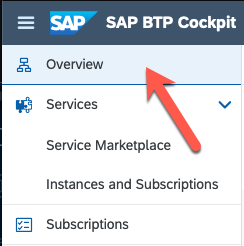
1. The SAP Cloud for Utilities Foundation Retailer URL you will be extending:

<https://edom-retailer-dev-dev-edom-retailer-srv.cfapps.eu10.hana.ondemand.com/>

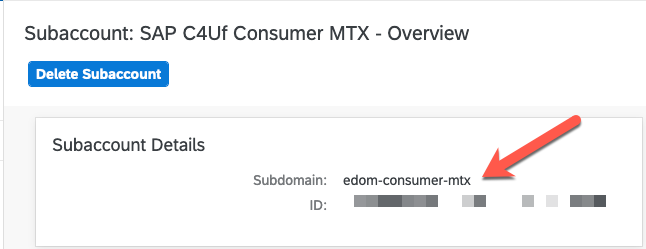
1. Your subaccount.

The subaccount in which you subscribed to the application (SAP Cloud for Utilities Foundation Retailer).

From the SAP BTP Cockpit, choose Overview.



The Subaccount Subdomain will be listed in the main page such as:



\*\*Note: If you have a custom domain setup …. ????

1. Create an empty directory in a preferred location on your computer.
2. Open VS Code and choose the newly created directoy. (File -> Open)
3. Prepare the final url:

* Create a new terminal in vscode
* Set the following env variable:

C4UDomain = “<https://edom-retailer-dev-dev-edom-retailer-srv.cfapps.eu10.hana.ondemand.com/>

SUBDOMAIN= your subdomain

e.g. SUBDOMAIN = edom-consumer-mtx

* The extension is always based on the deployed version of the existing application.

\*\*The deployment removes existing routes already created , they need to be recreated. (bug? Or saas-deploy)

\*\*The url for the server may change, Check with the development team about the change.

Based on our variables from above the complete domain is:

https://edom-consumer-mtx-edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com

Note: The domain-subdomain separator is a “.” in production not the “-“ as listed in the

line directly above.

1. Ensure you have been given the correct roles from the role template (at least ExtendCDS)

Please see above if this is not the case.

1. From the terminal execute the following command:

Reference for the cds extend command can be found [here](https://cap.cloud.sap/docs/guides/extensibility#start-extension-project)

cds extend -s edom-consumer-mtx <https://edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com>

As a security precaution you will be prompted for a passcode. The passcode comes from your

subaccount.

Therefore, in this case the passcode will be:

<https://edom-consumer-mtx.authentication.eu10.hana.ondemand.com/passcode>

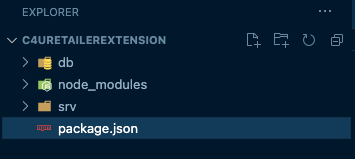
Note: You can shorten the workflow to [save the passcode](https://cap.cloud.sap/docs/guides/extensibility#save-authentication-token-for-simplified-workflow) to not have to query on every iteration.

1. If everything is successful you will see a message similar to the following:

Creating extension project for https://edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com, subdomain 'edom-consumer-mtx', in folder '/Users/ c4uRetailerExtension'

Retrieving files...

The following folder structure should be created:



Congratulations you are ready to start your new extension project!

1. It is a best practice to use some type of version control to track your extension project changes.

**The Extension Project Folder Structure**

Before starting our project it makes sense to understand the extension folder we have extended:

As you have noticed the folder structure consists of the following directories:

.

├── CAP Directory Structure

├── . db /

│ ├── This folder is where you put your extensions of entities and UIs (Fiori Elements). (link to fiori elements?)

│ └── (see db inside node\_modules for reference of models which can be extended)

├── node\_modules/

│ └── Expanding this folder you will notice an embedded structure:

├── This is the reference models and services exposed that you are able to extend./

│ └── base/

│ └── \_i18n : This contain the i18n of the provider./

│ ├── Open the file and see the i18n keys and values.

│ ├── db : The entities available to extend./

│ │ ├── (The following is an example of the exposed list)

│ │ ├── Please refer to released documentation to understand in detail the referenced

│ │ └── entities.(get link)/

│ │ └── models/

│ │ ├── alpha/

│ │ │ ├── configuration

│ │ │ ├── external

│ │ │ ├── mdiclient

│ │ │ └── serviceprovider

│ │ ├── extensions/

│ │ │ ├── dpp

│ │ │ ├── BusinessPartnerServiceProviderExtensions.cds

│ │ │ ├── CustomerOrderUtilitiesExtensions.cds

│ │ │ └── Index.cds

│ │ └── odm/

│ │ ├── billingaccount

│ │ ├── businesspartner

│ │ ├── common

│ │ ├── finance

│ │ ├── orgunit

│ │ ├── procurement

│ │ ├── product

│ │ ├── sales

│ │ └── index.cds

│ └── srv: The services that can be extended. (The following is an example of the exposed list)/

│ ├── alpha

│ ├── api

│ ├── dpp

│ ├── external

│ ├── monitoring

│ ├── cds-mtx-auth.cds

│ └── service.cds

├── srv

└── This folder is where you put your extensions of services. (see srv inside of node\_modules for reference)

**Note\*\***: There is also a node\_modules directory inside of \_base, this can be ignored.

Avoid running npm install otherwise it will delete the \_base folder.

**Creating the Extension Project (VS Code) – Part II (Simple Extension)**

Now that the directory structure has been created, we need to determine what extensions we would like to make.

The following are some of the boundary conditions we have to keep in mind before creating our extension:

1. Some entity names are reserved.

Prefixes of namespace extensions cannot be prefixed with com.sap.\* or sap.\*

1. In accordance with SAP customer extensions best practices, elements must be prefixed with either “Z\_” or “Z\_\_”
2. The maximum number of new entities (db) or new fields (srv – services) is 42.

We will start with a simple extension and then gradually build up. (Is this possible in CAP?? To reset? Sent email)

As mentioned previously, as with any development project there should be some type of version control.

Let’s get started,  
  
In this example we will extend CustomerOrder.

1. In VSCode create a file in the outer db directory and name it CustomerOrder\_db\_ext.cds (The naming of the file is not important however it must maintain the “.cds” extension)  
     
    Add the following to the file (keeping in mind the boundary conditions from above)

using sap.odm.sales from '\_base/db/models/odm/sales/Customer';

namespace Z\_edom.extension;

using {

cuid,

managed

} from '@sap/cds/common';

entity Z\_RefPOs : cuid, managed {

email : String;

firstName : String;

lastName : String;

}

1. Create a file in the outer srv directory and name it Customer\_srv\_ext.cds (Again the naming of the file is not important however remember to maintain the “.cds” extension.

Add the following to the file:

using from '\_base/srv/api/API\_EDOM\_RETAILER';

using {Z\_edom.extension as ext} from '../db/CustomerOrder\_db\_ext';

extend service API\_EDOM\_RETAILER with {

entity Z\_RefPOs as projection on ext.Z\_RefPOs;

}

1. Activate the extension.

We activate the extension by executing the following in the terminal:

cds activate --to https://edom-consumer-mtx-edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com -s edom-consumer-mtx  
  
\*\*Remember:

In this case the “–to” represents the full url (subdomain + domain)

the -s represents the subdomain

1. You may be prompted for a passcode.

The passcode is found on the subdomain. In this case it would be:  
  
 https://edom-consumer-mtx.authentication.eu10.hana.ondemand.com/passcode

1. The output from the command will be similar to:

Note: the authentication token is not saved by default. To save the token for later commands, please run `cds login`.

Activating extension located at /Users to URL https://edom-consumer-mtx-edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com

Activating extension...

Extension active

1. Congratulations the extension is now active!
2. As noted from the output above you can save the passcode token if you wish to continue modify the

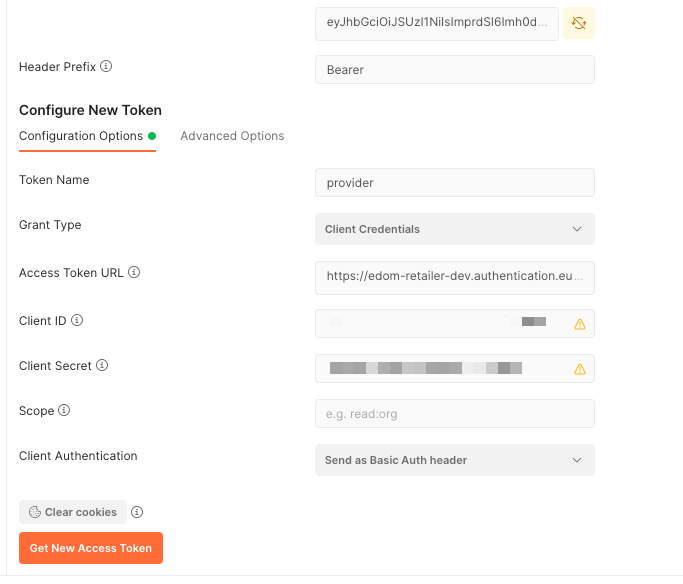
extension.

**Querying the extension (??? Info is from provider perspective because no app xsuaa exposed in client – follow up with CAP team about this)**

Once the extension has been activated, we need to verify if the data exists.

In order to be able to query from postman we need authorization to be able to view the data.

To get a new token from postman we need to query the authentication point in the following dialog:



Access Token URL:

This access token changes based on what you want to query.

For the provider domain it would be:

https://edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com/oauth/token

For the subdomain (consumer) it would be:

https://edom-consumer-mtx-edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com

The client ID and Client secret are retrieved from cloud foundry space.

Make sure to login into the correct space you are using:

Run the following command:

cf env (name of microservice)

The name of the microservice you are using to query.

For example:

If the query URL is the following:

https:// edom-consumer-mtx-edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com/api/v1/$metadata

We notice that the microservice that is being called is cpapp-srv.

Therefore, the cloud foundry command (cf) would be the following:

cf env cpapp-srv

It will retrieve the following json data:

We are interested in the xsuaa section:

"xsuaa": [

{

"clientid": "clientid",

"clientsecret": "client secret",

"url": “url” + “/oauth/token”

}

1. The grant type would be Client Credentials
2. Copy the ClientID and ClientSecret to the token retrieval dialog.
3. The authenticatin url would be the url above + ‘/oauth/token’
4. Enter the user credentials you are using.

SAP ID / Password in our case, customers may use their own IDP.

1. Scope if required.
2. Click “Get New Access Token”
3. Name and copy the token for your requests.

\*Note – The token has a time limit so you may need to generate a new one, depending on how much time has elapsed since the previous time you used it.

Examples:

**The provider should not have any extension**

1. Query the provider

GET:https://edom-retailer-dev-srv.cfapps.eu10.hana.ondemand.com/api/v1/$metadata

1. Check the results that the extension information is not there



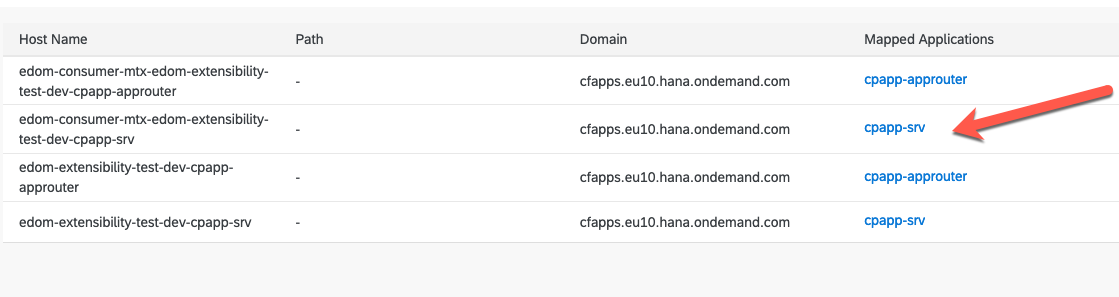
Perfect none there.

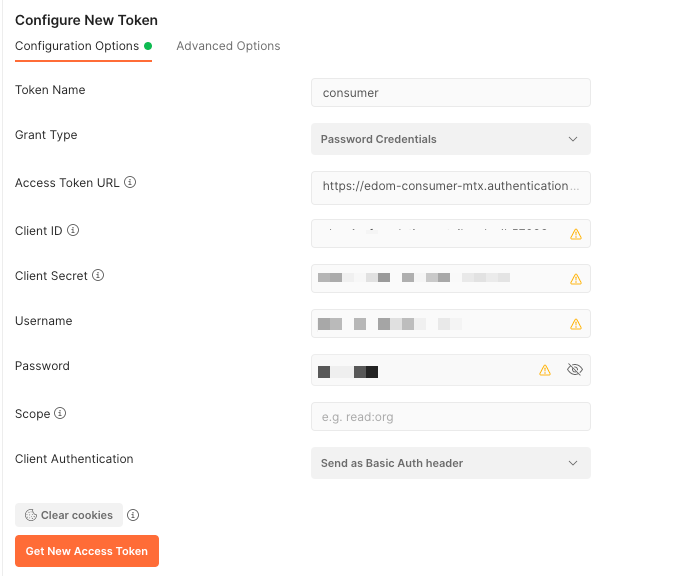
**The consumer being able to verify their own extensions**

In this case edom-consumer-mtx is the subdomain or consumer or extender

Edom-extenstibility-test-dev is the provider or base

1. If there isn’t a route create for the service (for the subdomain) this needs to be created. (



1. Use postman to retrieve the proper auth token.  
   The token is delivered by the subaccount  
     
   
2. Access to token uri is the subdomain:

https://edom-consumer-mtx.authentication.eu10.hana.ondemand.com/oauth/token

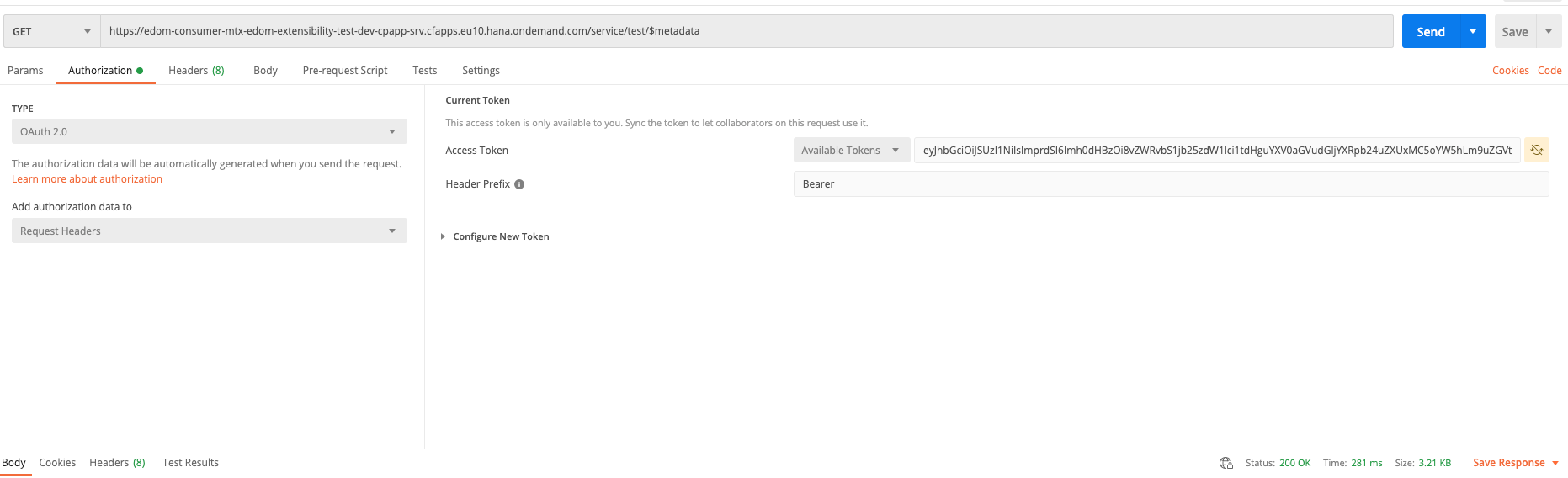
Client Id/Client secret are the env variables of the srv. Application in the xsuaa section

"clientid": "clientid",

"clientsecret": "clientsecret”

“url”: “url” + ‘/oauth/token’

Scope is not required  
  
3. Query the subaccount for the extension, we want to see the metadata  
  
https://edom-consumer-mtx-edom-extensibility-test-dev-cpapp-srv.cfapps.eu10.hana.ondemand.com/service/test/$metadata



1. View the extended data.  
   