Partner
Ecosystem
Success

Exercise 2 – Build a custom API for side-by-side extensibility

S/4HANA Cloud Developer Extensibility bootcamp

THE BEST RUN

TABLE OF CONTENTS

Introduction

With Developer Extensibility, it is now possible to create custom APIs in your S/4HANA Cloud system. These can be both inbound and outbound APIs and allow you to establish communication with external systems. As a developer, you use ABAP Development Tools for Eclipse (ADT) to implement such APIs. Keep in mind that whenever you are communicating with an external system, be it inbound or outbound communication, an administrator needs to perform the corresponding communication management to enable the usage of the desired APIs.

When developing custom **inbound** APIs, you have the option of using OData, plain HTTP or RFC communication.

Such custom APIs can be called remotely, for example from SAP Business Technology Platform. In a purely ABAP-based side-by-side extensibility scenario, your custom S/4HANA Cloud API could be consumed from an ABAP Cloud system on SAP BTP (i.e. SAP BTP, ABAP Environment).

Exercise scope

In this exercise, we will create a custom inbound OData API in an S/4HANA Cloud system. We will reuse the business object from the first day, i.e., the online shop, and create an OData service binding of type Web API on top of it. We will also create a custom communication scenario to expose our API.

In a real scenario, the development would be performed in the development tenant (client 080), while the communication management would be performed in the customizing tenant (client 100). This is because that is where we would normally call our custom API, in the customizing tenant containing more complete business configuration and data. For the purposes of this exercise, we will perform all activities in the development tenant.

We will consume the API from an ABAP system running on SAP BTP, i.e., an SAP BTP ABAP Environment service instance. We will use the API's metadata to create a service consumption model, which will allow us to call the API directly from ABAP coding. As a simple example, we will consume the API from a plain HTTP service, which will contain the custom coding in its handler class.

This HTTP service is a form of outbound communication, and thus also requires some communication management. We will therefore create another custom communication scenario to expose our outbound service.

Note:

A service consumption model analyzes an API's specifications and generates the ABAP development objects required to model the API. These allow for straightforward consumption of the API using ABAP coding. The service consumption model object page in ADT offers some code snippets for the different operations supported by the API. You can use these as a starting point for your own implementations.

Pre-requisites

ADT is required, as in Exercise 1. We will reuse the business object from the first exercise, so make sure that all objects up to the service definition are active and properly implemented.

Recommendations

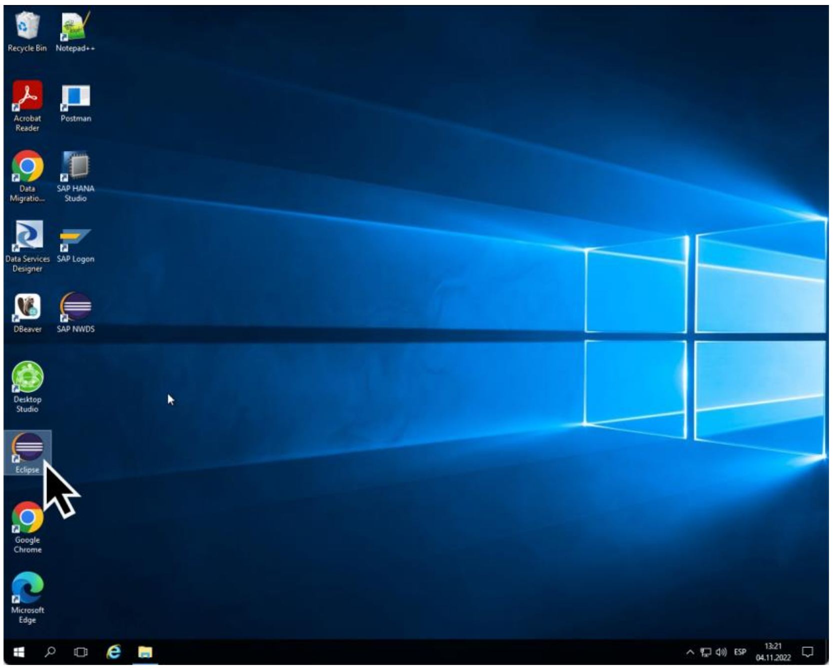
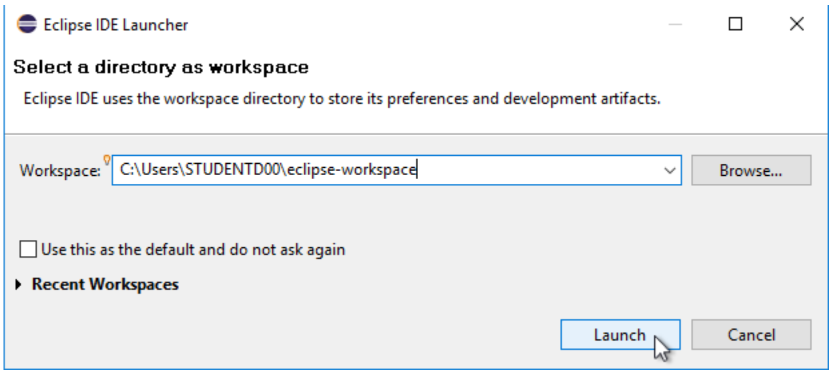
When naming objects, please make sure to include your ID in the name. This will uniquely identify your objects and prevent any confusion.
Example : Z_SHOP_API_<ID>

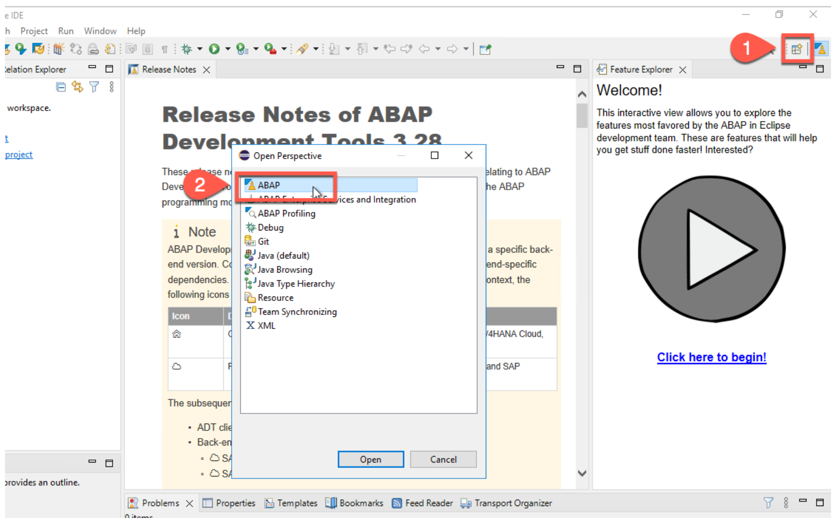
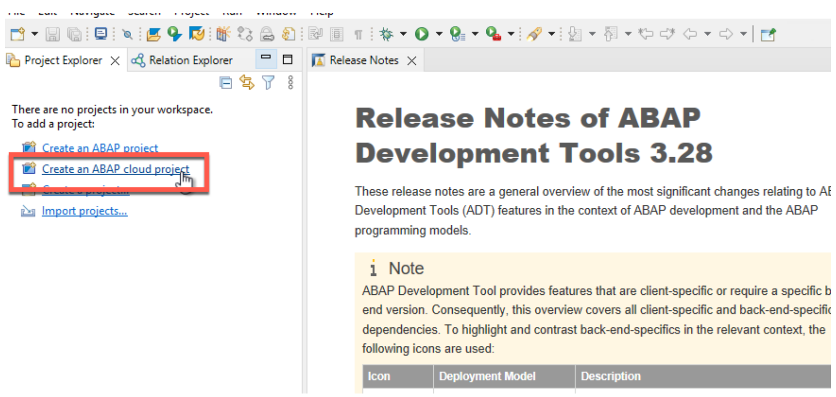
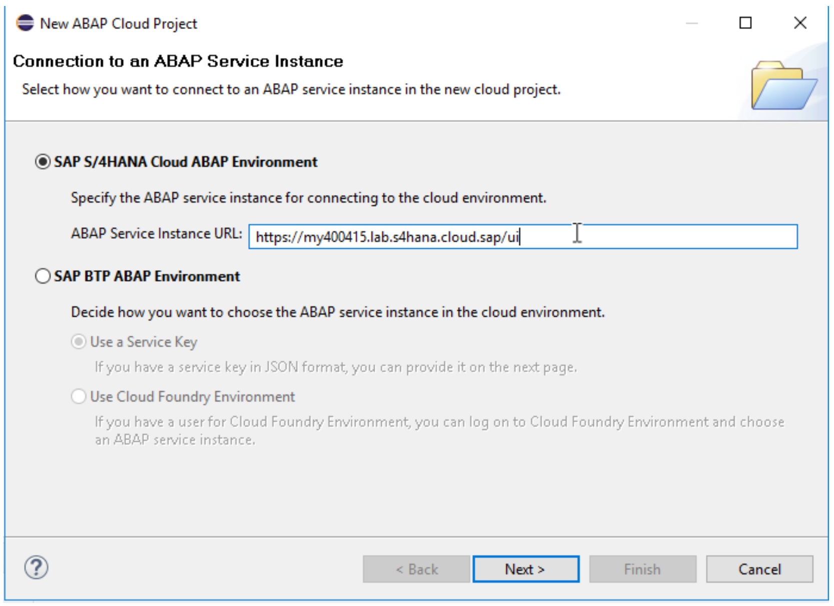
STEP 1 – CREATE API

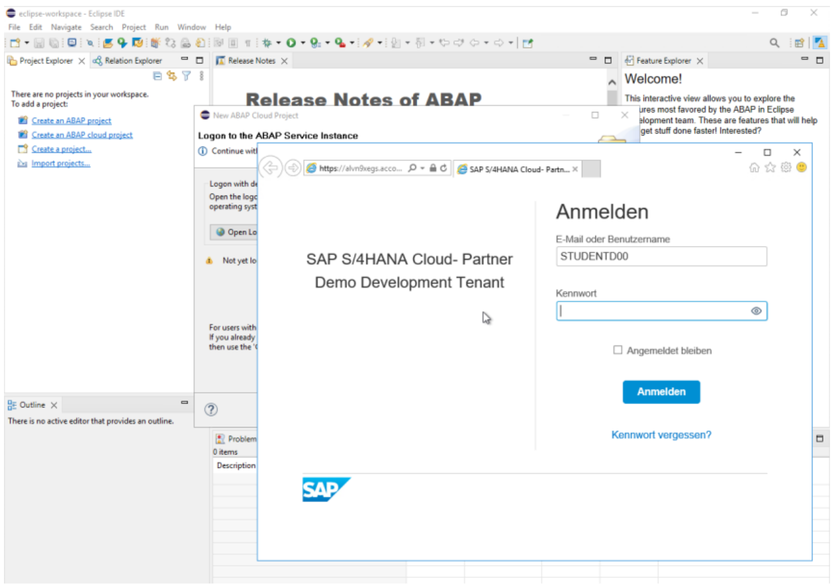
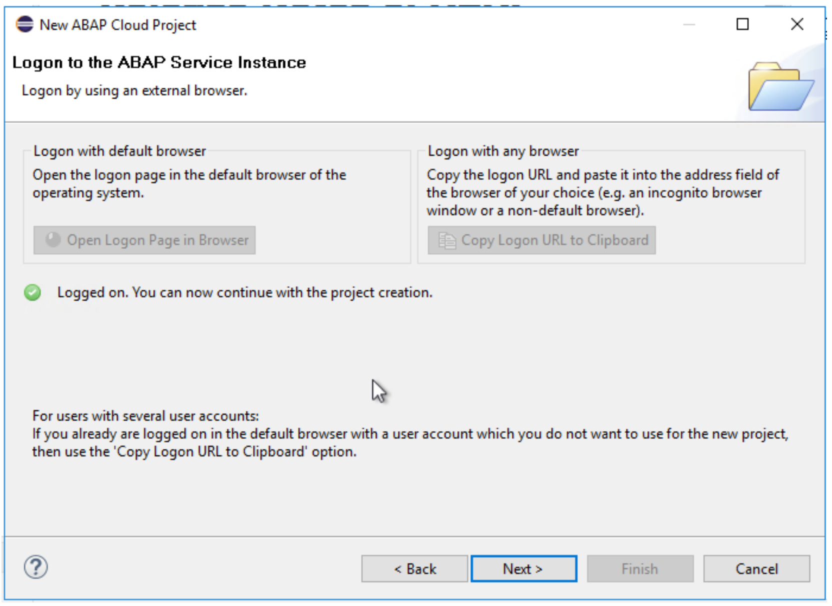
In S/4HANA Cloud, create and expose an inbound OData API

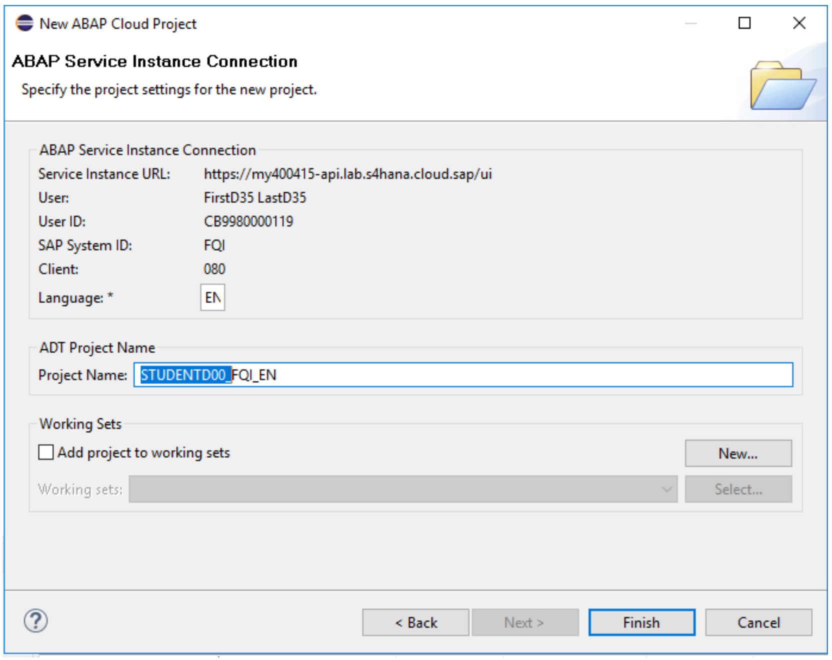
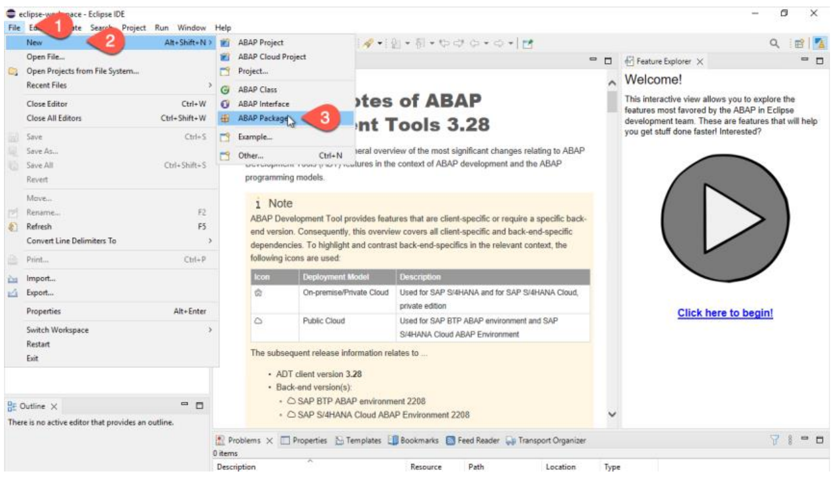
Step 1.1 – Create inbound API

As a developer, implement an OData service binding of type 'Web API' and create all required communication management development objects.

Description	Screenshot
Login to the S/4HANA Cloud Development Tenant via Eclipse To do this open Eclipse using the icon in the Remote Desktop.	 Recycle Bin Notepad++ Acrobat Reader Postman Data Migratio... SAP HANA Studio Data Services Designer SAP Logon DBeaver SAP NWDS Desktop Studio Eclipse Google Chrome Microsoft Edge
Create a new workspace and make sure you use your user folder. If the folder does not already exist, create a new one. Example: C:\Users\STUDENT<XXX>\eclipse-workspace where <XXX> is your ID (e.g. D01).	Eclipse IDE Launcher **Select a directory as workspace** Eclipse IDE uses the workspace directory to store its preferences and development artifacts. Workspace: C:\Users\STUDENTD00\eclipse-workspace ☐ Use this as the default and do not ask again **Recent Workspaces**

Description	Screenshot
Open the ABAP Perspective by clicking the “Open Perspective” icon on the top right and then selecting “ABAP” and finally pressing “Open”.	 1 2 Open Perspective ABAP ABAP Profiling Debug Git Java (default) Java Browsing Java Type Hierarchy Resource Team Synchronizing XML Open Cancel Release Notes of ABAP Development Tools 3.28 Feature Explorer Welcome! Click here to begin!
On the left side you have the Project Explorer bar. Click on “Create an ABAP Cloud Project”	 Project Explorer Relation Explorer Release Notes There are no projects in your workspace. To add a project: Create an ABAP project Create an ABAP cloud project Import projects... Release Notes of ABAP Development Tools 3.28
Select the first option “SAP S/4HANA Cloud ABAP Environment”. Use the S/4HANA Cloud URL available in the Cheat Sheet. This is connecting you to the Development Tenant. Make sure there are no spaces in between.	 New ABAP Cloud Project **Connection to an ABAP Service Instance** Select how you want to connect to an ABAP service instance in the new cloud project. ● **SAP S/4HANA Cloud ABAP Environment** Specify the ABAP service instance for connecting to the cloud environment. ABAP Service Instance URL: https://my400415.lab.s4hana.cloud.sap/ui ○ **SAP BTP ABAP Environment** Decide how you want to choose the ABAP service instance in the cloud environment. ○ Use a Service Key If you have a service key in JSON format, you can provide it on the next page. ○ Use Cloud Foundry Environment If you have a user for Cloud Foundry Environment, you can log on to Cloud Foundry Environment and choose an ABAP service instance.

Description	Screenshot
Open the system login page using “Open Logon Page in Browser”. Use your credentials from system “A” in your Cheat Sheet.	 Anmelden E-Mail oder Benutzername: STUDENTD00 Kennwort: ☐ Angemeldet bleiben Anmelden Kennwort vergessen? SAP S/4HANA Cloud- Partner Demo Development Tenant
Once the login is done successfully, you can close the browser window and you should see a success message in Eclipse “Logged on….”. Click “Next”.	 New ABAP Cloud Project **Logon to the ABAP Service Instance** Logon by using an external browser. Logon with default browser: Open the logon page in the default browser of the operating system. [Open Logon Page in Browser] Logon with any browser: Copy the logon URL and paste it into the address field of the browser of your choice (e.g. an incognito browser window or a non-default browser). [Copy Logon URL to Clipboard] ✓ Logged on. You can now continue with the project creation. For users with several user accounts: If you already are logged on in the default browser with a user account which you do not want to use for the new project, then use the 'Copy Logon URL to Clipboard' option. < Back · Next > · Finish · Cancel

Description	Screenshot
Define the local project name. Use your user ID as a prefix to distinguish your project. Click “Finish”.	
Select File >> New >> ABAP Package	

New ABAP Cloud Project

ABAP Service Instance Connection

Specify the project settings for the new project.

ABAP Service Instance Connection

Service Instance URL:	https://my400415-api.lab.s4hana.cloud.sap/ui
User:	FirstD35 LastD35
User ID:	CB9980000119
SAP System ID:	FQI
Client:	080
Language: *	EN

ADT Project Name

Project Name: STUDENTD00_FQI_EN

Working Sets

[ ] Add project to working sets

Working sets:

otes of ABAP
nt Tools 3.28

neral overview of the most significant changes relating to ABAP Development Tools (ADT) features in the context of ABAP development and the ABAP programming models.

Note

ABAP Development Tool provides features that are client-specific or require a specific back-end version. Consequently, this overview covers all client-specific and back-end-specific dependencies. To highlight and contrast back-end-specifics in the relevant context, the following icons are used:

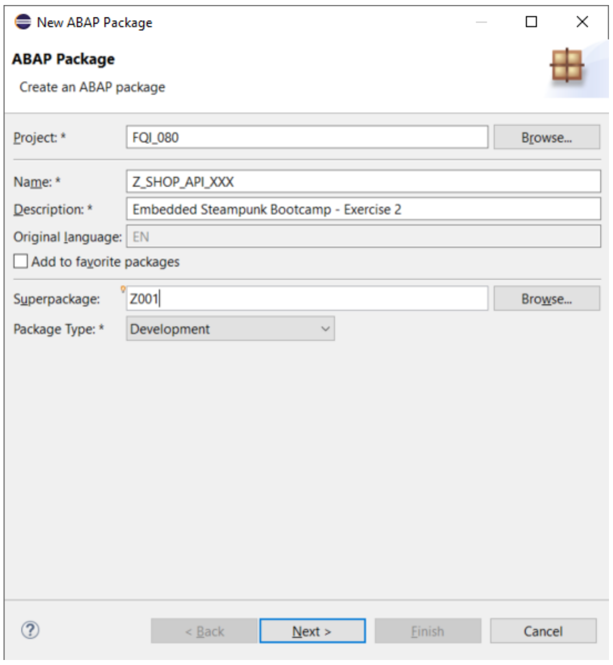
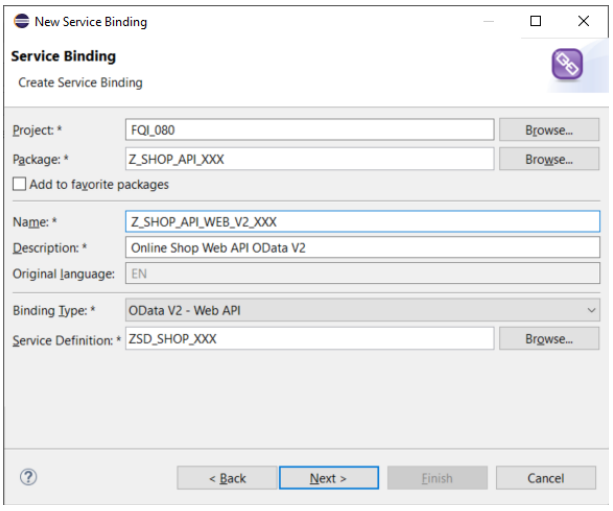
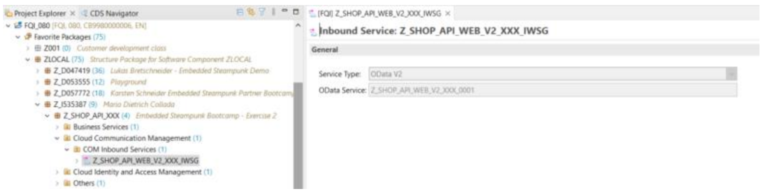
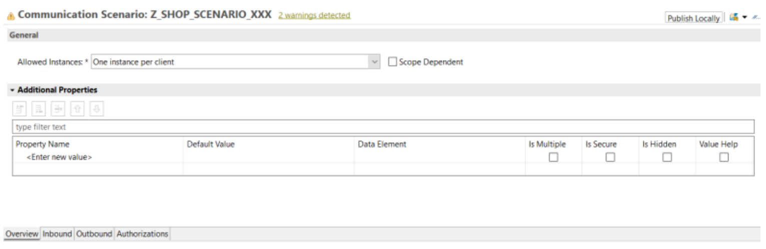
Icon	Deployment Model	Description
⌂	On-premise/Private Cloud	Used for SAP S/4HANA and for SAP S/4HANA Cloud, private edition
☁	Public Cloud	Used for SAP BTP ABAP environment and SAP S/4HANA Cloud ABAP Environment

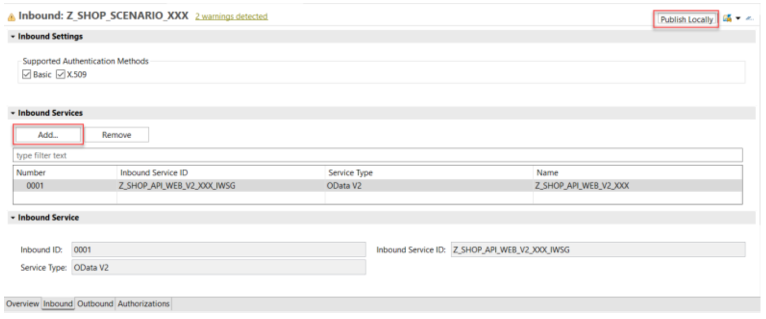
The subsequent release information relates to ...

- ADT client version 3.28
- Back-end version(s):
  - ☁ SAP BTP ABAP environment 2208
  - ☁ SAP S/4HANA Cloud ABAP Environment 2208

This interactive view allows you to explore the features most favored by the ABAP in Eclipse development team. These are features that will help you get stuff done faster! Interested?

There is no active editor that provides an outline.

Description	Screenshot
Create a new package for this exercise	 New ABAP Package **ABAP Package** Create an ABAP package Project: * FQI_080 Name: * Z_SHOP_API_XXX Description: * Embedded Steampunk Bootcamp - Exercise 2 Original language: EN ☐ Add to favorite packages Superpackage: Z001 Package Type: * Development < Back · Next > · Finish · Cancel
Create a new *Service Binding* object of type *Web API* Choose the service definition created in earlier exercises as your referenced object In this exercise we are using OData V2, but you could also use OData V4. Activate and publish the service binding.	 New Service Binding **Service Binding** Create Service Binding Project: * FQI_080 Package: * Z_SHOP_API_XXX ☐ Add to favorite packages Name: * Z_SHOP_API_WEB_V2_XXX Description: * Online Shop Web API OData V2 Original language: EN Binding Type: * OData V2 - Web API Service Definition: * ZSD_SHOP_XXX < Back · Next > · Finish · Cancel
Make sure that a *COM Inbound Service* object was created under *Cloud Communication Management*. This is auto generated.	 Project Explorer · CDS Navigator FQI_080 [FQI, 080, CB9980000006, EN] Favorite Packages (75) Z001 (0) Customer development class ZLOCAL (75) Structure Package for Software Component ZLOCAL Z_D047419 (36) Lukas Bretschneider - Embedded Steampunk Demo Z_D053555 (12) Playground Z_D057772 (18) Karsten Schneider Embedded Steampunk Partner Bootcamp Z_I535387 (9) Mario Dietrich Collado Z_SHOP_API_XXX (4) Embedded Steampunk Bootcamp - Exercise 2 Business Services (1) Cloud Communication Management (1) COM Inbound Services (1) Z_SHOP_API_WEB_V2_XXX_IWSG Cloud Identity and Access Management (1) Others (1) [FQI] Z_SHOP_API_WEB_V2_XXX_IWSG **Inbound Service: Z_SHOP_API_WEB_V2_XXX_IWSG** General Service Type: OData V2 OData Service: Z_SHOP_API_WEB_V2_XXX_0001
Create a custom *Communication Scenario* object Z_COM_SHOP_SCENARIO_XXX Provide description: Scenario for inbound calls to online shop API Set *Allowed Instances* to one instance per client, as we only want to set up one communication arrangement.	 **Communication Scenario: Z_SHOP_SCENARIO_XXX** 2 warnings detected · Publish Locally General Allowed Instances: * One instance per client · ☐ Scope Dependent Additional Properties type filter text Property Name · Default Value · Data Element · Is Multiple · Is Secure · Is Hidden · Value Help <Enter new value> ☐ ☐ ☐ ☐ Overview · Inbound · Outbound · Authorizations

Description	Screenshot
In the *Inbound* tab, add the Inbound Service from the earlier step to your custom communication scenario. Make sure that Basic Authentication is enabled, as that is what we are going to use (to keep things simple). Save the *Communication Scenario.* Go back to the overview tab and Publish Locally.	

Inbound: Z_SHOP_SCENARIO_XXX 2 warnings detected

Inbound Settings

Supported Authentication Methods: ✓ Basic ✓ X.509

Inbound Services

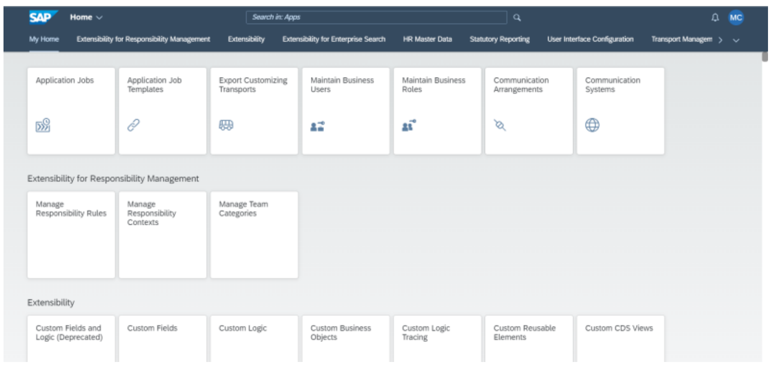
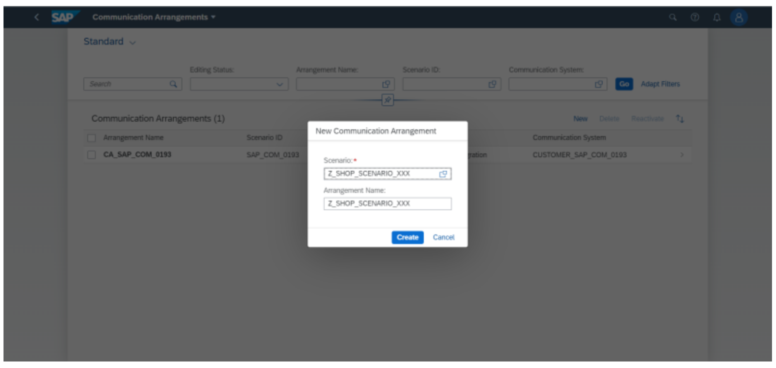
Number	Inbound Service ID	Service Type	Name
0001	Z_SHOP_API_WEB_V2_XXX_IWSG	OData V2	Z_SHOP_API_WEB_V2_XXX

Inbound Service

Inbound ID: 0001
Inbound Service ID: Z_SHOP_API_WEB_V2_XXX_IWSG
Service Type: OData V2

Step 1.2 – Expose inbound API

As an administrator, perform the required communication management to expose the inbound API.

Description	Screenshot
Login to the S/4HANA Cloud Development Tenant (080) Launchpad via your browser. You can find the link in the cheat sheet.	
Open the *Communication Arrangements* Fiori app. Create a new communication arrangement, using the previously created custom *Communication Scenario* as your reference. **Please do not modify any existing communication arrangements or other artifacts that do not belong to you!**	

SAP
Home
Search in: Apps
MC
My Home
Extensibility for Responsibility Management
Extensibility
Extensibility for Enterprise Search
HR Master Data
Statutory Reporting
User Interface Configuration
Transport Managem
Application Jobs
Application Job Templates
Export Customizing Transports
Maintain Business Users
Maintain Business Roles
Communication Arrangements
Communication Systems
Extensibility for Responsibility Management
Manage Responsibility Rules
Manage Responsibility Contexts
Manage Team Categories
Extensibility
Custom Fields and Logic (Deprecated)
Custom Fields
Custom Logic
Custom Business Objects
Custom Logic Tracing
Custom Reusable Elements
Custom CDS Views

Communication Arrangements (1)

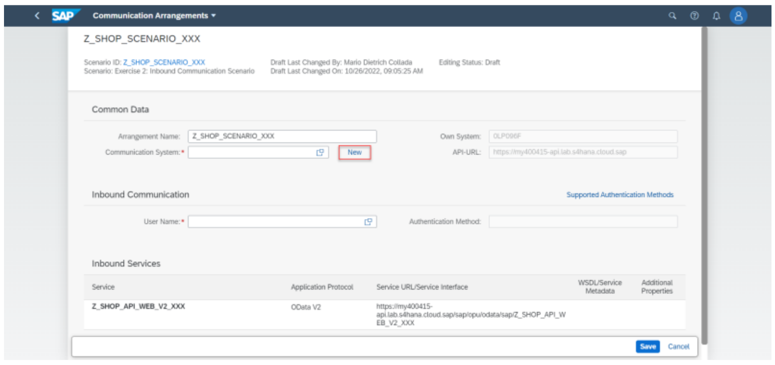
Arrangement Name	Scenario ID		Communication System
CA_SAP_COM_0193	SAP_COM_0193	ration	CUSTOMER_SAP_COM_0193

New Communication Arrangement

Scenario:* Z_SHOP_SCENARIO_XXX
Arrangement Name: Z_SHOP_SCENARIO_XXX

Press the *New* button next to the *Communication System* field. Choose a System ID and a System Name.

After pressing *Create*, you will be forwarded to the configuration page of your new communication system. Choose a system ID and system name.



Z_SHOP_SCENARIO_XXX

Scenario ID: Z_SHOP_SCENARIO_XXX
Scenario: Exercise 2: Inbound Communication Scenario
Draft Last Changed By: Mario Dietrich Collada
Draft Last Changed On: 10/26/2022, 09:05:25 AM
Editing Status: Draft

Common Data

Arrangement Name: Z_SHOP_SCENARIO_XXX
Communication System:*
Own System: 0LP096F
API-URL: https://my400415-api.lab.s4hana.cloud.sap

Inbound Communication

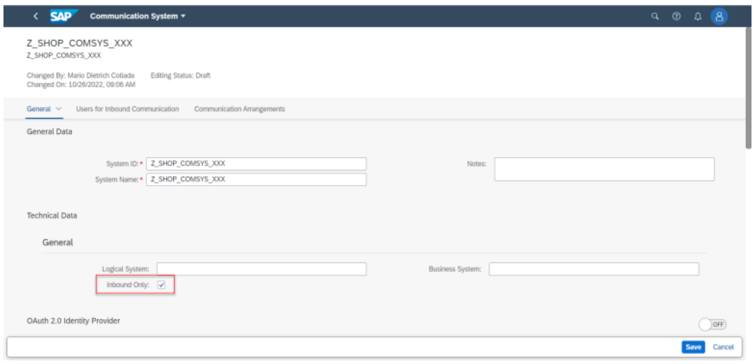
User Name:*
Authentication Method:

Inbound Services

Service	Application Protocol	Service URL/Service Interface	WSDL/Service Metadata	Additional Properties
Z_SHOP_API_WEB_V2_XXX	OData V2	https://my400415-api.lab.s4hana.cloud.sap/sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX		

Check the *Inbound Only* checkbox under *Technical Data – General.*

Since we are only exposing an inbound API, we do not need to specify any host name or such.



Z_SHOP_COMSYS_XXX
Z_SHOP_COMSYS_XXX

Changed By: Mario Dietrich Collada
Changed On: 10/26/2022, 09:06 AM
Editing Status: Draft

General Data

System ID:* Z_SHOP_COMSYS_XXX
System Name:* Z_SHOP_COMSYS_XXX
Notes:

Technical Data

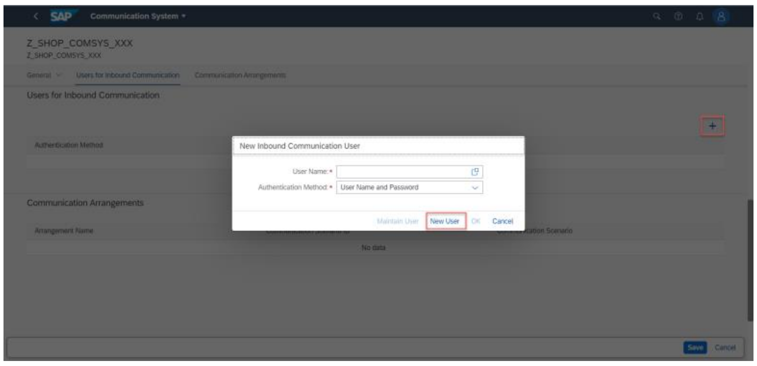
General

Logical System:
Business System:
Inbound Only: ✓

OAuth 2.0 Identity Provider OFF

Under *Users for Inbound Communication*, add a new technical user via the **+** button.

Since we do not yet have a suitable user, we can create one directly from the current screen using the *New User* button.



Z_SHOP_COMSYS_XXX
Z_SHOP_COMSYS_XXX

Users for Inbound Communication

Authentication Method

Communication Arrangements

Arrangement Name

No data

New Inbound Communication User

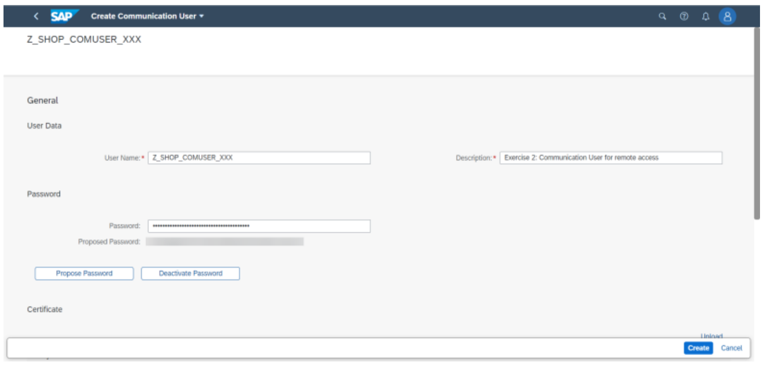
User Name:*
Authentication Method:* User Name and Password

Specify a user name and a description.

Specify a password or generate one using the *Propose Password* button. Make sure to store the password, as we will need it later.

Press *Create* to save your settings.

Note:
Communication users can authenticate via password or via certificate. We will be using password authentication in this exercise.



Z_SHOP_COMUSER_XXX

General

User Data

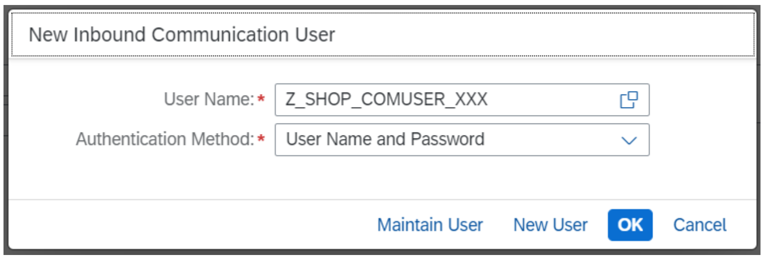
User Name:* Z_SHOP_COMUSER_XXX
Description:* Exercise 2: Communication User for remote access

Password

Password: ••
Proposed Password:

Certificate

As you navigate back to the communication system, confirm that you want to add the newly created communication user by pressing *OK*



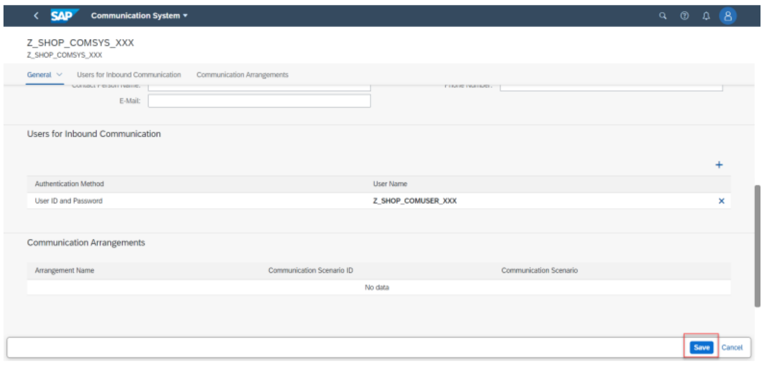
New Inbound Communication User

User Name:* Z_SHOP_COMUSER_XXX

Authentication Method:* User Name and Password

Save your communication system.

You will be redirected back to the original communication arrangement.



Z_SHOP_COMSYS_XXX
Z_SHOP_COMSYS_XXX

E-Mail:

Users for Inbound Communication

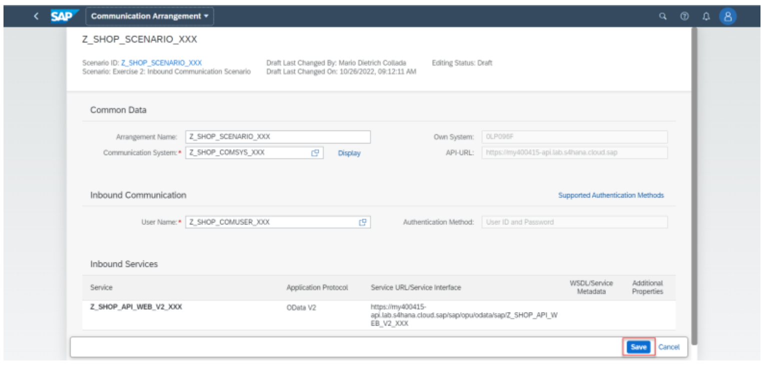
Authentication Method	User Name
User ID and Password	Z_SHOP_COMUSER_XXX

Communication Arrangements

Arrangement Name	Communication Scenario ID	Communication Scenario
	No data	

Save your communication arrangement.

The previously created communication user will automatically be used. If you were to maintain multiple users in your communication system, you could select one at the arrangement level.



Z_SHOP_SCENARIO_XXX

Scenario ID: Z_SHOP_SCENARIO_XXX
Scenario: Exercise 2: Inbound Communication Scenario
Draft Last Changed By: Mario Dietrich Collada
Draft Last Changed On: 10/26/2022, 09:12:11 AM
Editing Status: Draft

Common Data

Arrangement Name: Z_SHOP_SCENARIO_XXX
Communication System:* Z_SHOP_COMSYS_XXX
Own System: 0LP096F
API-URL: https://my400415-api.lab.s4hana.cloud.sap

Inbound Communication

User Name:* Z_SHOP_COMUSER_XXX
Authentication Method: User ID and Password

Inbound Services

Service	Application Protocol	Service URL/Service Interface	WSDL/Service Metadata	Additional Properties
Z_SHOP_API_WEB_V2_XXX	OData V2	https://my400415-api.lab.s4hana.cloud.sap/sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX		

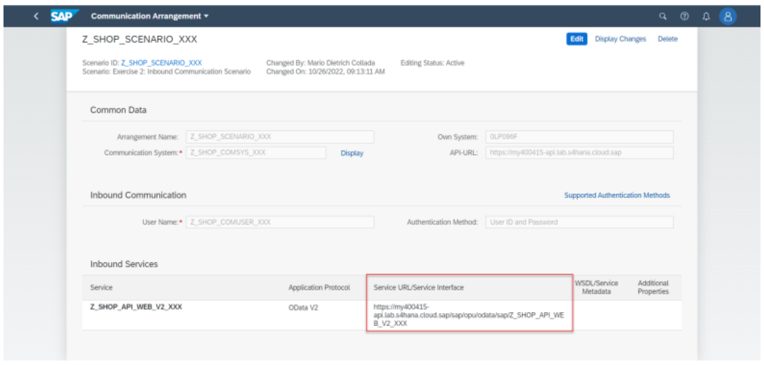
Before we move on, let’s download the service metadata of the inbound service.

Retrieve the metadata as follows:

1. Copy the *Service URL* shown in your arrangement
2. Remove the *“-api”* substring
3. Append “*/$metadata*” to the URL
4. Open the resulting URL in a browser
5. Save the displayed file using the context menu → *Save As…*

Example (Actual URL can be obtained as per above steps): https://my400788.lab.s4hana.cloud.sap/sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX/$metadata

If there is an option to download the service metadata directly from the communication arrangement app, you could also do this.



Z_SHOP_SCENARIO_XXX

Scenario ID: Z_SHOP_SCENARIO_XXX
Scenario: Exercise 2: Inbound Communication Scenario
Changed By: Mario Dietrich Collada
Changed On: 10/26/2022, 09:13:11 AM
Editing Status: Active

Common Data

Arrangement Name: Z_SHOP_SCENARIO_XXX
Communication System:* Z_SHOP_COMSYS_XXX
Own System: 0LP096F
API-URL: https://my400415-api.lab.s4hana.cloud.sap

Inbound Communication

User Name:* Z_SHOP_COMUSER_XXX
Authentication Method: User ID and Password

Inbound Services

Service	Application Protocol	Service URL/Service Interface	WSDL/Service Metadata	Additional Properties
Z_SHOP_API_WEB_V2_XXX	OData V2	https://my400415-api.lab.s4hana.cloud.sap/sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX		

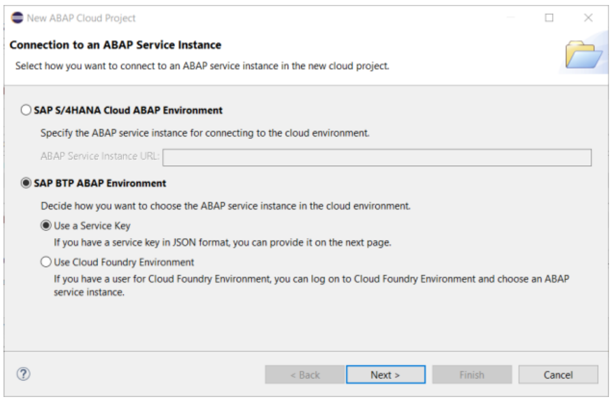
This metadata file will allow us to comfortably consume the service from SAP BTP. Make sure to note down the Service URL of your inbound service as well, as we will need it at a later point.	

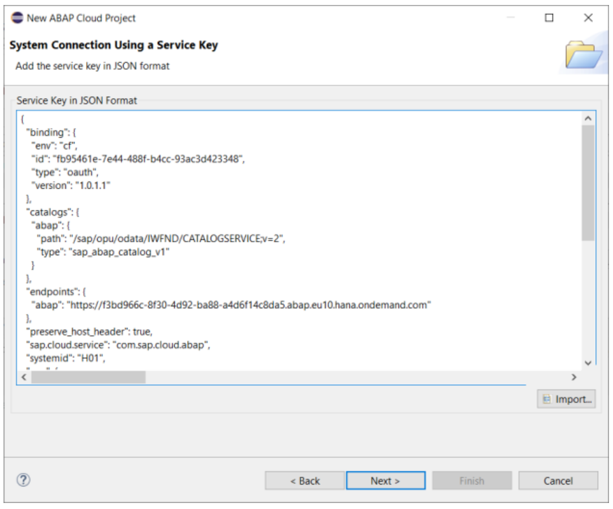
STEP 2 – CONSUME API

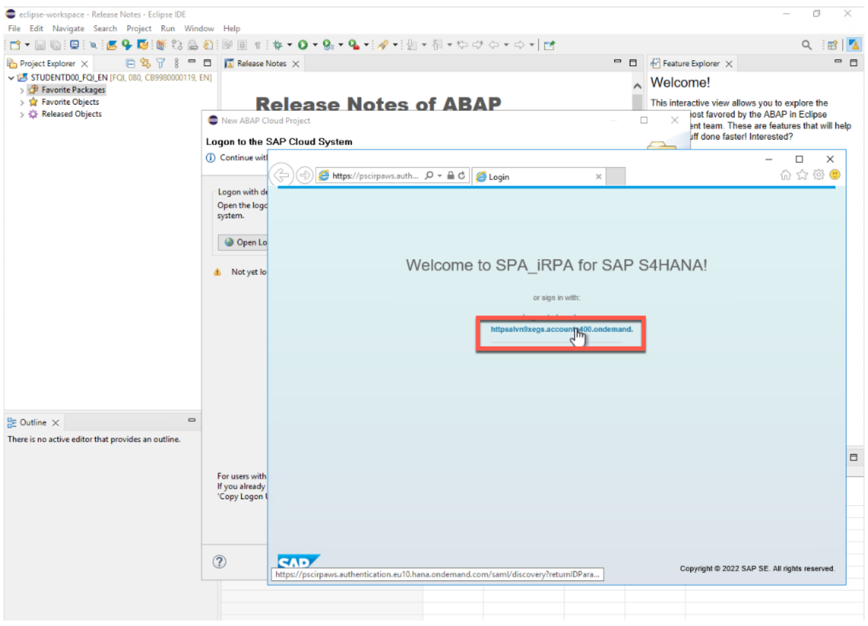
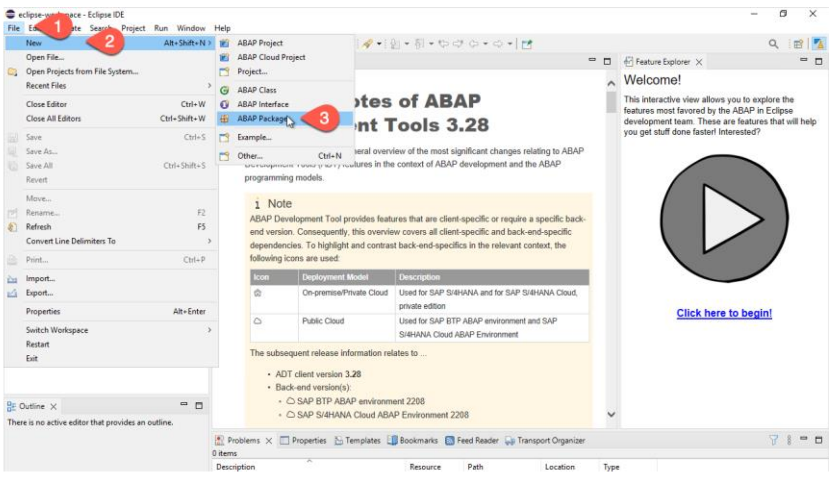
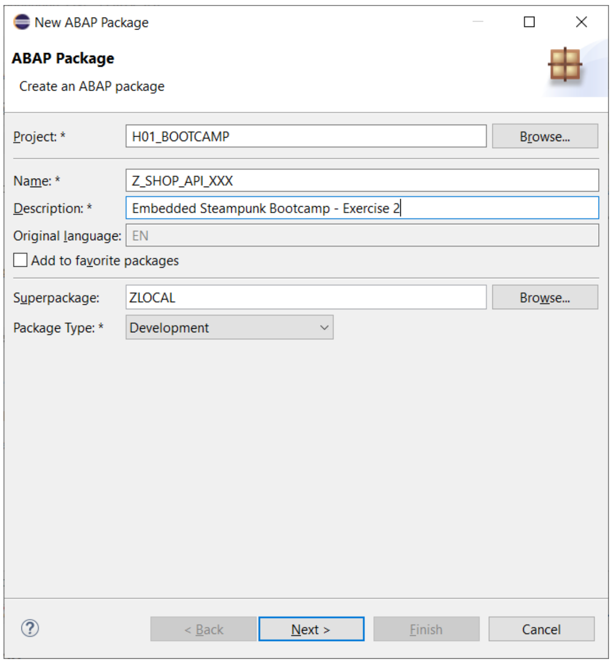
Consume your custom OData API in SAP BTP ABAP Environment

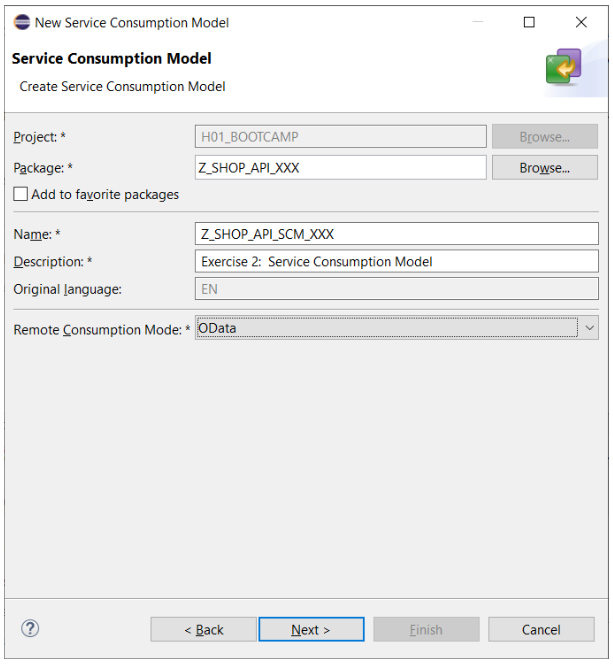
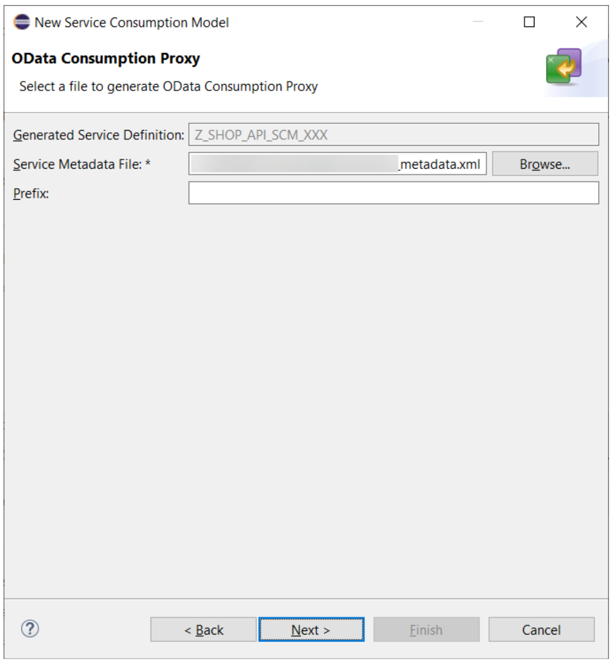
Step 2.1 – Create Service Consumption Model

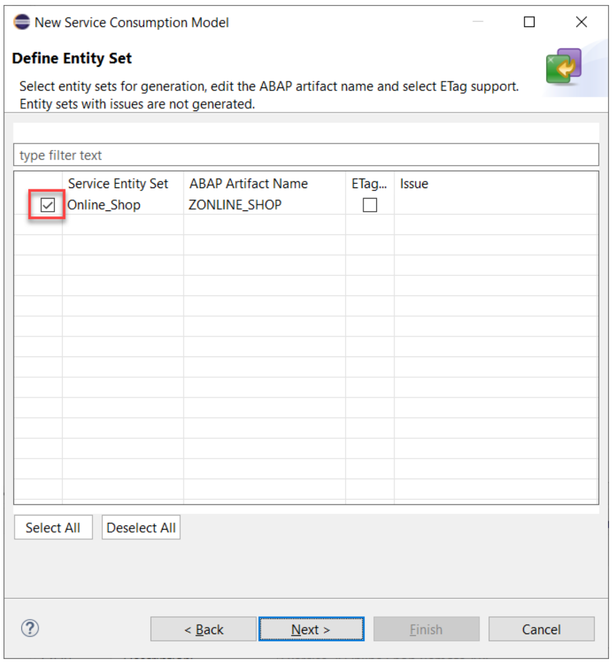
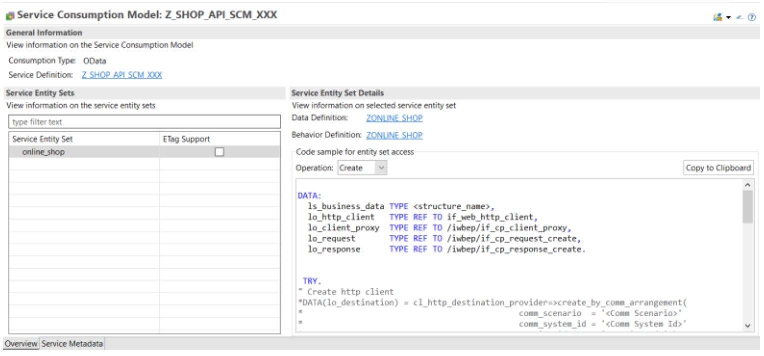
As a developer on SAP BTP, implement a service consumption model to consume the external OData API.

Description	Screenshot
Login to the SAP BTP ABAP Environment instance via Eclipse. Start by creating a new ABAP Cloud Project and choosing the option *SAP BTP ABAP Environment – Use a Service Key*	 New ABAP Cloud Project **Connection to an ABAP Service Instance** Select how you want to connect to an ABAP service instance in the new cloud project. SAP S/4HANA Cloud ABAP Environment Specify the ABAP service instance for connecting to the cloud environment. ABAP Service Instance URL: [✓ SAP BTP ABAP Environment] Decide how you want to choose the ABAP service instance in the cloud environment. [✓ Use a Service Key] If you have a service key in JSON format, you can provide it on the next page. Use Cloud Foundry Environment If you have a user for Cloud Foundry Environment, you can log on to Cloud Foundry Environment and choose an ABAP service instance. < Back Next > Finish Cancel

Paste in the service key of the ABAP Environment instance.
 	
You can find it at the end of this document HERE.
	
	New ABAP Cloud Project
	System Connection Using a Service Key
	Add the service key in JSON format
	Service Key in JSON Format
	`{`
	`"binding": {`
	`"env": "cf",`
	`"id": "fb95461e-7e44-488f-b4cc-93ac3d423348",`
	`"type": "oauth",`
	`"version": "1.0.1.1"`
	`},`
	`"catalogs": {`
	`"abap": {`
	`"path": "/sap/opu/odata/IWFND/CATALOGSERVICE;v=2",`
	`"type": "sap_abap_catalog_v1"`
	`}`
	`},`
	`"endpoints": {`
	`"abap": "https://f3bd966c-8f30-4d92-ba88-a4d6f14c8da5.abap.eu10.hana.ondemand.com"`
	`},`
	`"preserve_host_header": true,`
	`"sap.cloud.service": "com.sap.cloud.abap",`
	`"systemid": "H01",`
	Import...
	< Back Next > Finish Cancel

Description	Screenshot
Proceed to the next step and choose *Open Logon Page in Browser*. Select the second option called "httpsalvn9xegs…." Enter the credentials from your cheat sheet. Once logged in successfully close the browser window. Follow the remaining steps of the wizard.	 New ABAP Cloud Project Logon to the SAP Cloud System Welcome to SPA_iRPA for SAP S4HANA! or sign in with: httpsalvn9xegs.accounts400.ondemand.
Select File >> New >> ABAP Package	 File > New > ABAP Package
Create a new package for this exercise under structure package ZLOCAL.	 New ABAP Package **ABAP Package** Create an ABAP package Project: * H01_BOOTCAMP Name: * Z_SHOP_API_XXX Description: * Embedded Steampunk Bootcamp - Exercise 2 Original language: EN ☐ Add to favorite packages Superpackage: ZLOCAL Package Type: * Development < Back Next > Finish Cancel

Description	Screenshot
Create a new *Service Consumption Model* object. Specify a name, a description and choose *OData* as the *Remote Consumption Mode*. Press *Next*.	 New Service Consumption Model **Service Consumption Model** Create Service Consumption Model Project: * H01_BOOTCAMP Package: * Z_SHOP_API_XXX Add to favorite packages Name: * Z_SHOP_API_SCM_XXX Description: * Exercise 2: Service Consumption Model Original language: EN Remote Consumption Mode: * OData Browse... Browse... < Back Next > Finish Cancel
Locate the metadata file that you previously downloaded. Use it to define the service consumption model. Press *Next*.	 New Service Consumption Model **OData Consumption Proxy** Select a file to generate OData Consumption Proxy Generated Service Definition: Z_SHOP_API_SCM_XXX Service Metadata File: * metadata.xml Prefix:

Description	Screenshot
The creation wizard will show you a list of the entities defined in the service. Make sure that the relevant entities are selected. In this case we just have the one entry. Press *Next*. The wizard will show you a list of the objects that shall be generated. You can then use these objects to consume the external API directly from your ABAP code. **Hint: If the generated ABAP Artifact Name is not easy to read, you could edit this to provide a unique name** You can now proceed with the creation.	
Make sure to activate your *Service Consumption Model* once it is created.	

New Service Consumption Model

Define Entity Set

Select entity sets for generation, edit the ABAP artifact name and select ETag support.
Entity sets with issues are not generated.

type filter text

	Service Entity Set	ABAP Artifact Name	ETag...	Issue
☑	Online_Shop	ZONLINE_SHOP	☐	

Select All | Deselect All

< Back | Next > | Finish | Cancel

Service Consumption Model: Z_SHOP_API_SCM_XXX

General Information
View information on the Service Consumption Model
Consumption Type: OData
Service Definition: Z_SHOP_API_SCM_XXX

Service Entity Sets
View information on the service entity sets
type filter text

Service Entity Set	ETag Support
online_shop	☐

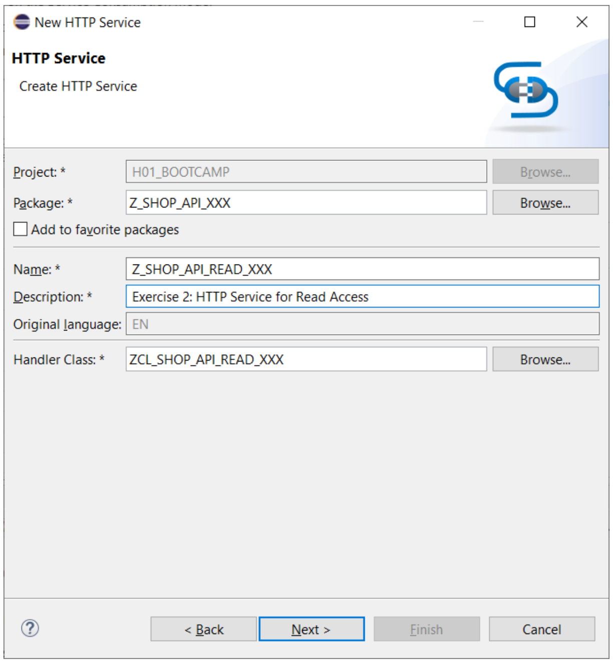
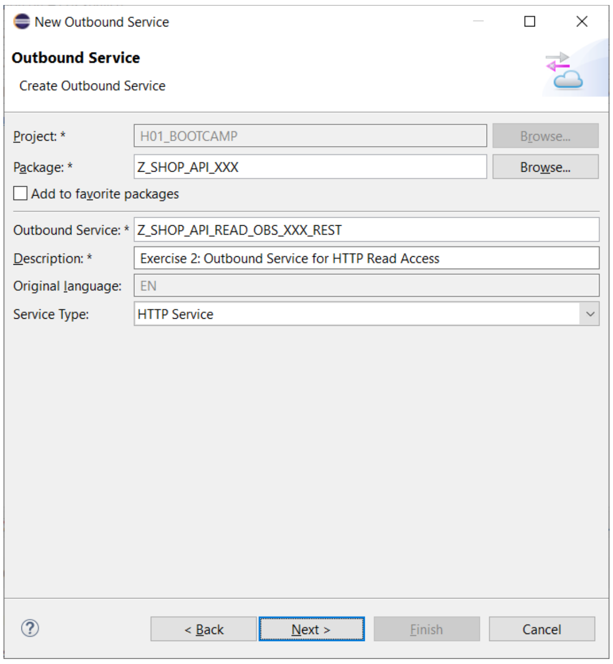
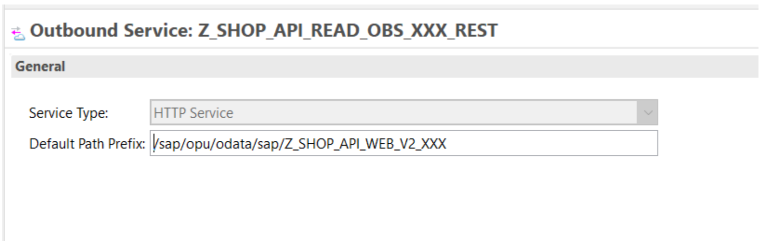
Service Entity Set Details
View information on selected service entity set
Data Definition: ZONLINE_SHOP
Behavior Definition: ZONLINE_SHOP
Code sample for entity set access
Operation: Create
Copy to Clipboard

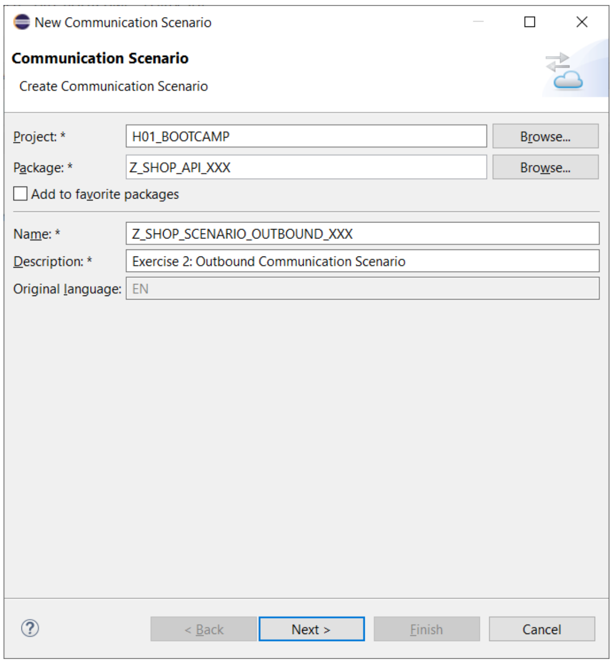
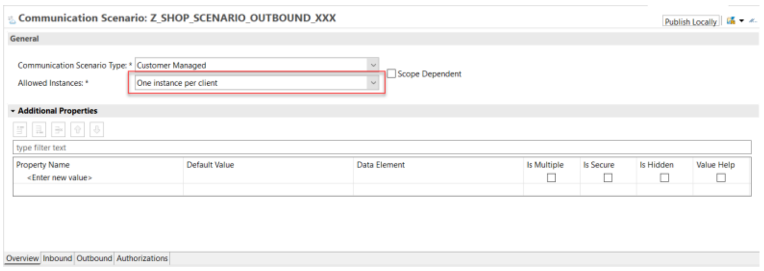
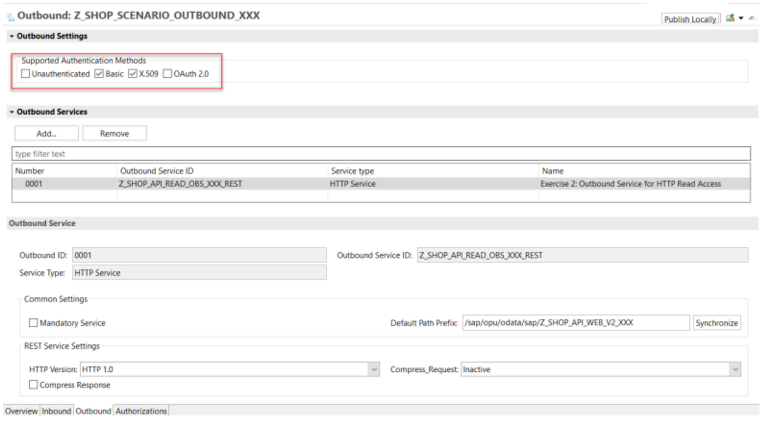
```
DATA:
  ls_business_data TYPE <structure_name>,
  lo_http_client   TYPE REF TO if_web_http_client,
  lo_client_proxy  TYPE REF TO /iwbep/if_cp_client_proxy,
  lo_request       TYPE REF TO /iwbep/if_cp_request_create,
  lo_response      TYPE REF TO /iwbep/if_cp_response_create.


 TRY.
" Create http client
*DATA(lo_destination) = cl_http_destination_provider=>create_by_comm_arrangement(
*                                                 comm_scenario  = '<Comm Scenario>'
*                                                 comm_system_id = '<Comm System Id>'
```

Overview | Service Metadata

Step 2.2 – Create HTTP service

As a developer on SAP BTP, implement a plain HTTP service which consumes the previously created service consumption model.

Description	Screenshot
Create a new *HTTP Service* object. Choose a name and a description, as well as a name for the handler class. The class will be automatically generated. Save and activate.	 New HTTP Service **HTTP Service** Create HTTP Service Project: * H01_BOOTCAMP Package: * Z_SHOP_API_XXX ☐ Add to favorite packages Name: * Z_SHOP_API_READ_XXX Description: * Exercise 2: HTTP Service for Read Access Original language: EN Handler Class: * ZCL_SHOP_API_READ_XXX
Create a new *Outbound Service* object of type *HTTP Service*.	 New Outbound Service **Outbound Service** Create Outbound Service Project: * H01_BOOTCAMP Package: * Z_SHOP_API_XXX ☐ Add to favorite packages Outbound Service: * Z_SHOP_API_READ_OBS_XXX_REST Description: * Exercise 2: Outbound Service for HTTP Read Access Original language: EN Service Type: HTTP Service
Specify the default path prefix for your service. You can get this information from the communication arrangement on S/4HANA Cloud side. Example: /sap/opu/odata/sap/<binding-name> Save the object.	 **Outbound Service: Z_SHOP_API_READ_OBS_XXX_REST** **General** Service Type: HTTP Service Default Path Prefix: /sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX

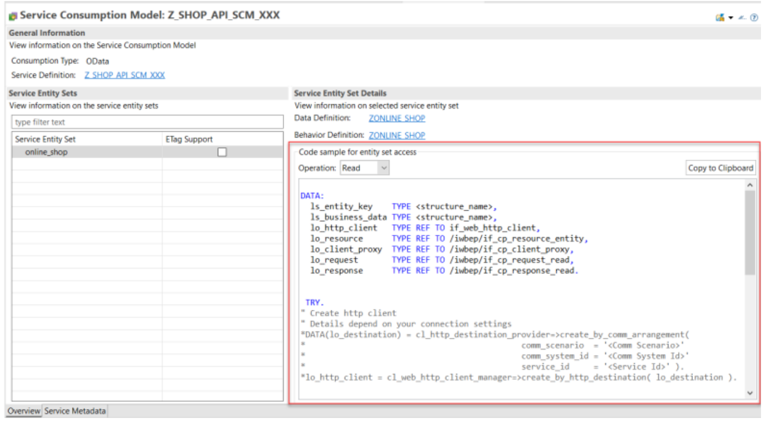
Description	Screenshot
Create a custom *Communication Scenario* object.	 New Communication Scenario **Communication Scenario** Create Communication Scenario Project: * H01_BOOTCAMP Package: * Z_SHOP_API_XXX ☐ Add to favorite packages Name: * Z_SHOP_SCENARIO_OUTBOUND_XXX Description: * Exercise 2: Outbound Communication Scenario Original language: EN
Set *Allowed Instances* to one instance per client, as we only want to set up one communication arrangement.	 Communication Scenario: Z_SHOP_SCENARIO_OUTBOUND_XXX **General** Communication Scenario Type: * Customer Managed Allowed Instances: * One instance per client ☐ Scope Dependent **Additional Properties** type filter text Property Name | Default Value | Data Element | Is Multiple | Is Secure | Is Hidden | Value Help <Enter new value> | | | ☐ | ☐ | ☐ | ☐
Under *Outbound*, add the previously created *Outbound Service.* Make sure that *Basic Authentication* is enabled. Save and publish the scenario locally.	 Outbound: Z_SHOP_SCENARIO_OUTBOUND_XXX **Outbound Settings** Supported Authentication Methods ☐ Unauthenticated ☑ Basic ☑ X.509 ☐ OAuth 2.0 **Outbound Services** type filter text Number | Outbound Service ID | Service type | Name 0001 | Z_SHOP_API_READ_OBS_XXX_REST | HTTP Service | Exercise 2: Outbound Service for HTTP Read Access **Outbound Service** Outbound ID: 0001 Outbound Service ID: Z_SHOP_API_READ_OBS_XXX_REST Service Type: HTTP Service Common Settings ☐ Mandatory Service Default Path Prefix: /sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX REST Service Settings HTTP Version: HTTP 1.0 Compress_Request: Inactive ☐ Compress Response

Description | **Screenshot**

Now we will implement the HTTP service.

Navigate to the generated handler class, where we will start to implement the `if_http_service_extension~handle_request` method. The name of the handler class was specified when we created the HTTP service in a previous step.

We can get a good starting point by using the code snippets provided in the service consumption model object. Select the *Read* operation and copy the provided code over into your method.



Service Consumption Model: Z_SHOP_API_SCM_XXX

General Information
View information on the Service Consumption Model
Consumption Type: OData
Service Definition: Z_SHOP_API_SCM_XXX

Service Entity Sets
View information on the service entity sets

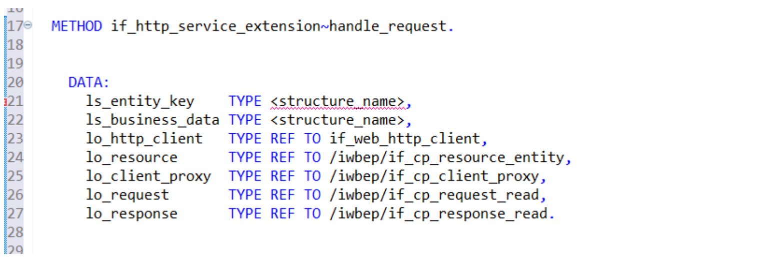
Service Entity Set	ETag Support
online_shop	☐

Service Entity Set Details
View information on selected service entity set
Data Definition: ZONLINE_SHOP
Behavior Definition: ZONLINE_SHOP

Code sample for entity set access
Operation: Read

```
DATA:
  ls_entity_key    TYPE <structure_name>,
  ls_business_data TYPE <structure_name>,
  lo_http_client   TYPE REF TO if_web_http_client,
  lo_resource      TYPE REF TO /iwbep/if_cp_resource_entity,
  lo_client_proxy  TYPE REF TO /iwbep/if_cp_client_proxy,
  lo_request       TYPE REF TO /iwbep/if_cp_request_read,
  lo_response      TYPE REF TO /iwbep/if_cp_response_read.


 TRY.
" Create http client
" Details depend on your connection settings
*DATA(lo_destination) = cl_http_destination_provider=>create_by_comm_arrangement(
*                                              comm_scenario  = '<Comm Scenario>'
*                                              comm_system_id = '<Comm System Id>'
*                                              service_id     = '<Service Id>' ).
*lo_http_client = cl_web_http_client_manager=>create_by_http_destination( lo_destination ).
```

Replace instances of <structure_name> with the name of the CDS view (Data Definition) that was generated as part of the consumption model.
This name is shown on the service consumption model object page



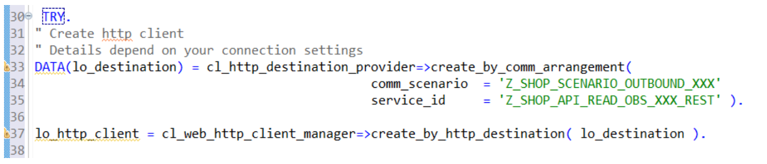
```
  METHOD if_http_service_extension~handle_request.


    DATA:
      ls_entity_key     TYPE <structure_name>,
      ls_business_data  TYPE <structure_name>,
      lo_http_client    TYPE REF TO if_web_http_client,
      lo_resource       TYPE REF TO /iwbep/if_cp_resource_entity,
      lo_client_proxy   TYPE REF TO /iwbep/if_cp_client_proxy,
      lo_request        TYPE REF TO /iwbep/if_cp_request_read,
      lo_response       TYPE REF TO /iwbep/if_cp_response_read.
```

Uncomment the method call `cl_http_destination_provider=>create_by_comm_arrangement` and adjust the parameters:

- Specify the name of the custom *Communication Scenario* that you just created
- Specify the name of the *Outbound Service* that you just created
- We do not need to specify a communication system

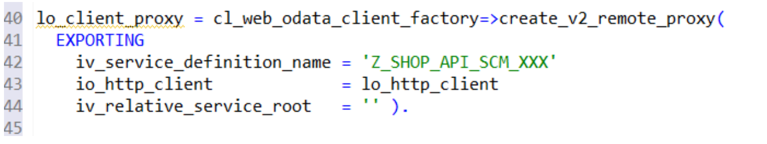
Using this information, the correct service endpoint will be derived at runtime, provided that a corresponding communication arrangement and system are maintained (see later steps).

Uncomment the `cl_web_http_client_manager=>create_by_http_destination` method call.



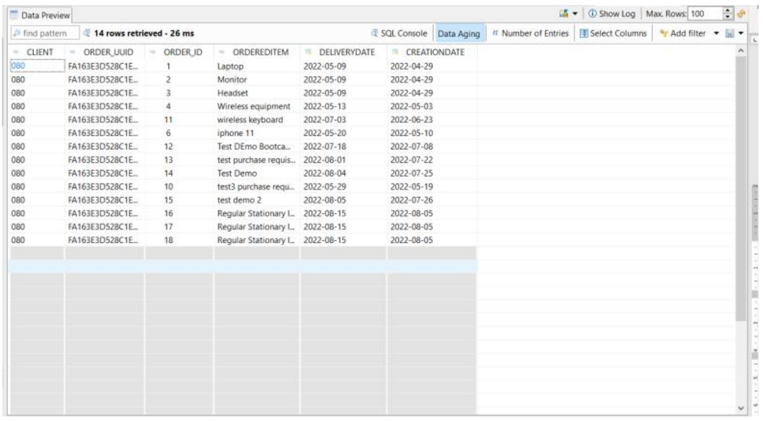
```
TRY.
" Create http client
" Details depend on your connection settings
DATA(lo_destination) = cl_http_destination_provider=>create_by_comm_arrangement(
                                              comm_scenario  = 'Z_SHOP_SCENARIO_OUTBOUND_XXX'
                                              service_id     = 'Z_SHOP_API_READ_OBS_XXX_REST' ).

lo_http_client = cl_web_http_client_manager=>create_by_http_destination( lo_destination ).
```

Replace the string `<service_root>` with an empty string. We have already specified the path in the outbound service object, hence there is no need to maintain it here.



```
lo_client_proxy = cl_web_odata_client_factory=>create_v2_remote_proxy(
  EXPORTING
    iv_service_definition_name = 'Z_SHOP_API_SCM_XXX'
    io_http_client             = lo_http_client
    iv_relative_service_root   = '' ).
```

Description	Screenshot

Specify the key of the entry that you want to read. For this exercise, we will simply hardcode a single (existing) value.
 	
You can view the UUIDs of all entries by opening the underlying database table (from the first exercise) in your S/4HANA Cloud project and using the *Data Preview* functionality.
	`" Set entity key`
	`ls_entity_key = VALUE #(`
	`order_uuid  = 'FA163E3D528C1EDCB1F59A2102C43B33' ).`
	

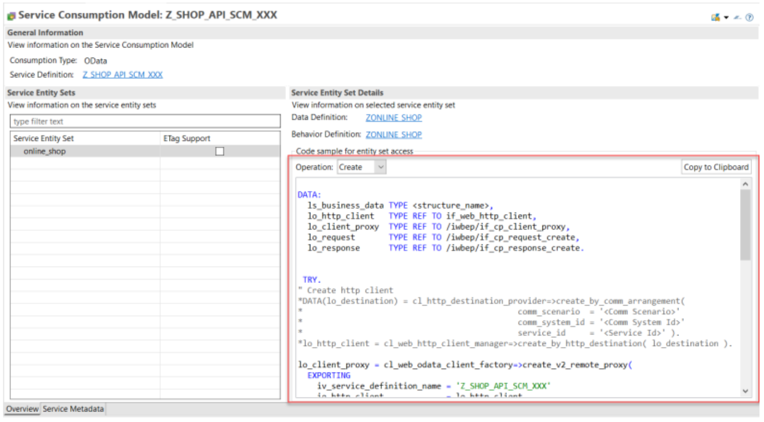
Use the results from the read operation to pass some meaningful information to the end user.
 	
We will write the Order ID and the generated purchase requisition ID to the response body using the appropriate setter method.
	`DATA: result type string.`
	`result = | OrderID: { ls_business_data-Order_Id }, Purchase Requisition ID: { ls_business_data-Purchasereqn  } |.`
	`response->set_text( result ).`

Data Preview

Show Log Max. Rows: 100

find pattern 14 rows retrieved - 26 ms SQL Console Data Aging Number of Entries Select Columns Add filter

CLIENT	ORDER_UUID	ORDER_ID	ORDEREDITEM	DELIVERYDATE	CREATIONDATE
080	FA163E3D528C1E...	1	Laptop	2022-05-09	2022-04-29
080	FA163E3D528C1E...	2	Monitor	2022-05-09	2022-04-29
080	FA163E3D528C1E...	3	Headset	2022-05-09	2022-04-29
080	FA163E3D528C1E...	4	Wireless equipment	2022-05-13	2022-05-03
080	FA163E3D528C1E...	11	wireless keyboard	2022-07-03	2022-06-23
080	FA163E3D528C1E...	6	iphone 11	2022-05-20	2022-05-10
080	FA163E3D528C1E...	12	Test DEmo Bootca...	2022-07-18	2022-07-08
080	FA163E3D528C1E...	13	test purchase requis...	2022-08-01	2022-07-22
080	FA163E3D528C1E...	14	Test Demo	2022-08-04	2022-07-25
080	FA163E3D528C1E...	10	test3 purchase requ...	2022-05-29	2022-05-19
080	FA163E3D528C1E...	15	test demo 2	2022-08-05	2022-07-26
080	FA163E3D528C1E...	16	Regular Stationary I...	2022-08-15	2022-08-05
080	FA163E3D528C1E...	17	Regular Stationary I...	2022-08-15	2022-08-05
080	FA163E3D528C1E...	18	Regular Stationary I...	2022-08-15	2022-08-05

Description	Screenshot
Catch exceptions `cx_http_dest_provider_error` and `cx_web_http_client_error`. Make sure that all caught exceptions write something descriptive to the response text.	CATCH /iwbep/cx_cp_remote INTO DATA(lx_remote). " Handle remote Exception " It contains details about the problems of your http(s) connection response->set_text(| Remote Error: { lx_remote->get_longtext( ) } |). CATCH /iwbep/cx_gateway INTO DATA(lx_gateway). " Handle Exception response->set_text(| Gateway Error: { lx_gateway->get_longtext( ) } |). CATCH cx_http_dest_provider_error INTO DATA(lx_destination). "handle exception response->set_text(| Destination Error: { lx_destination->get_longtext( ) } |). CATCH cx_web_http_client_error INTO DATA(lx_http). "handle exception response->set_text(| HTTP Error: { lx_http->get_longtext( ) } |).
Optional If you want, you can also create additional HTTP services to perform different operations. For example, you can create another service to perform a *Create* operation. You will need to implement a handler class for each HTTP service. You can use the different code snippets provided in the service consumption model. Remember to create additional *Outbound Services* for each additional HTTP service and add them to your custom communication scenario.	

Service Consumption Model: Z_SHOP_API_SCM_XXX

General Information

View information on the Service Consumption Model

Consumption Type: OData

Service Definition: Z_SHOP_API_SCM_XXX

Service Entity Sets

View information on the service entity sets

Service Entity Set	ETag Support
online_shop	☐

Service Entity Set Details

View information on selected service entity set

Data Definition: ZONLINE_SHOP

Behavior Definition: ZONLINE_SHOP

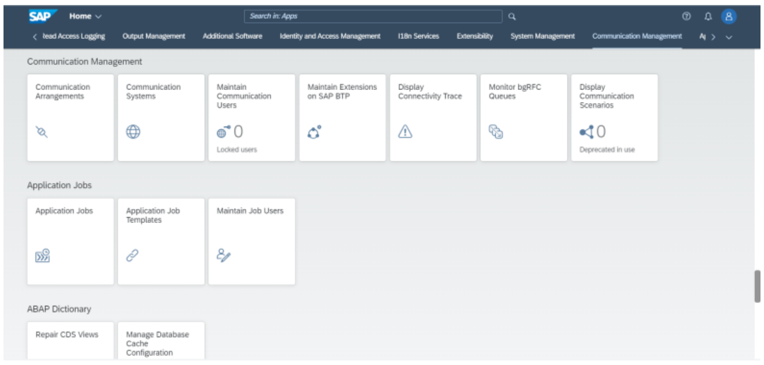
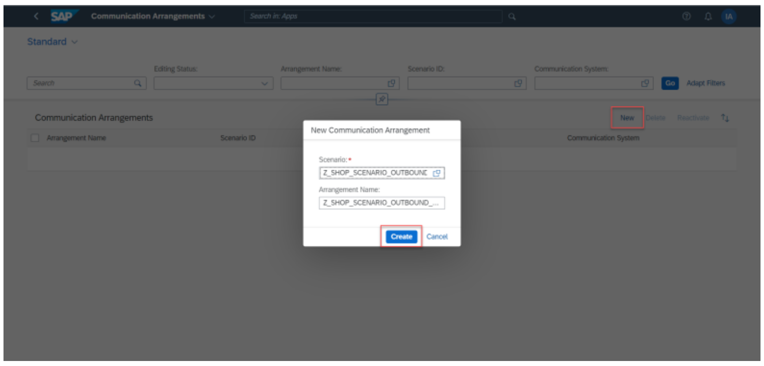
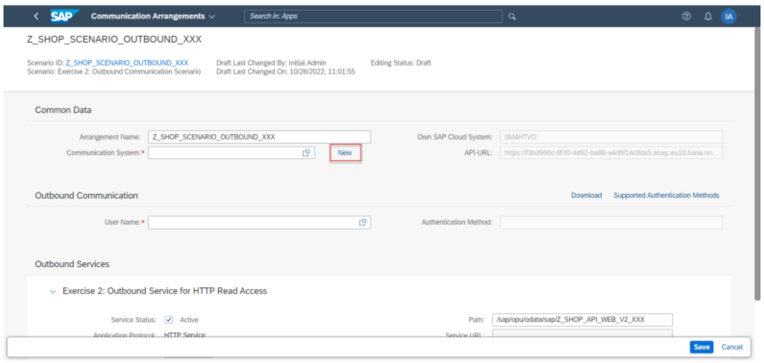
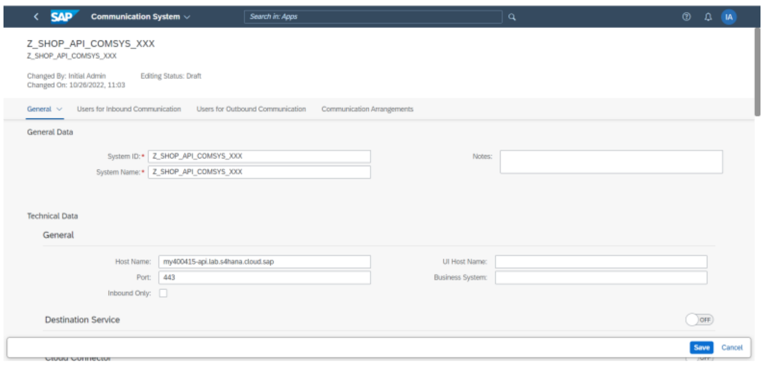
Code sample for entity set access

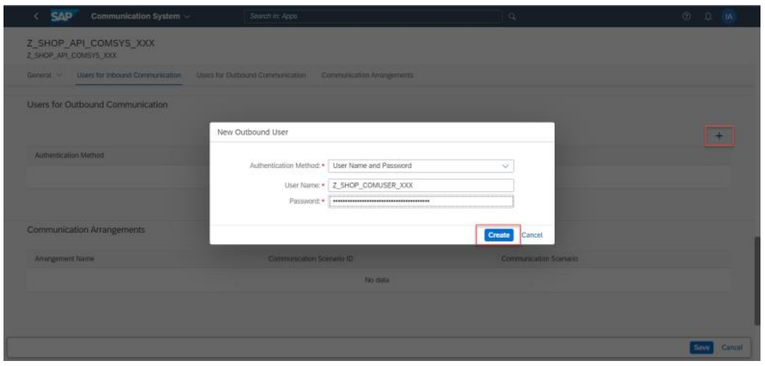
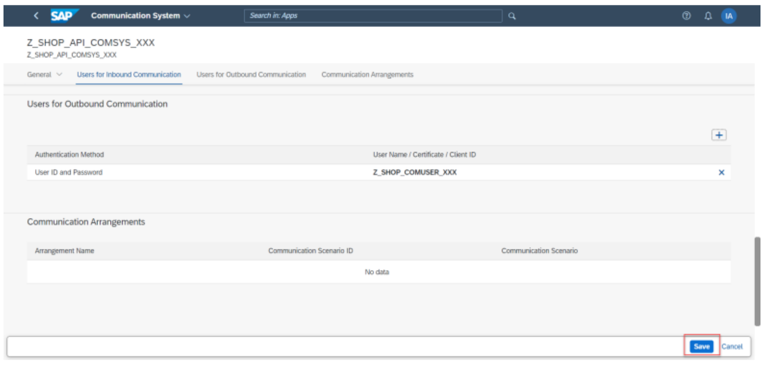
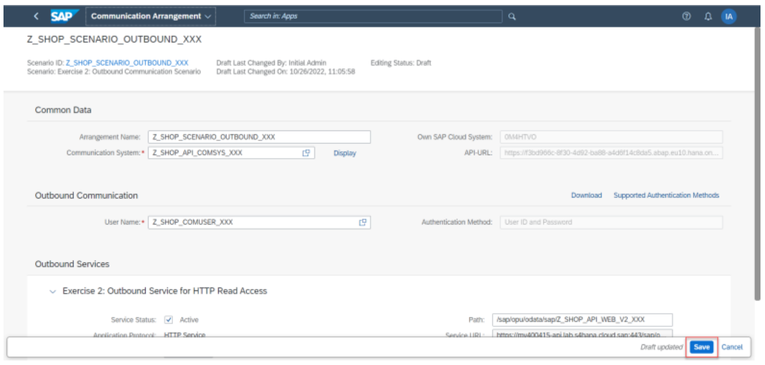
Operation: Create

```
DATA:
  ls_business_data TYPE <structure_name>,
  lo_http_client   TYPE REF TO if_web_http_client,
  lo_client_proxy  TYPE REF TO /iwbep/if_cp_client_proxy,
  lo_request       TYPE REF TO /iwbep/if_cp_request_create,
  lo_response      TYPE REF TO /iwbep/if_cp_response_create.


 TRY.
" Create http client
*DATA(lo_destination) = cl_http_destination_provider=>create_by_comm_arrangement(
*                                             comm_scenario  = '<Comm Scenario>'
*                                             comm_system_id = '<Comm System Id>'
*                                             service_id     = '<Service Id>' ).
*lo_http_client = cl_web_http_client_manager=>create_by_http_destination( lo_destination ).

lo_client_proxy = cl_web_odata_client_factory=>create_v2_remote_proxy(
  EXPORTING
    iv_service_definition_name = 'Z_SHOP_API_SCM_XXX'
```

Step 2.3 – Expose HTTP service

As an administrator on SAP BTP, perform the outbound communication management required to consume the external OData API.

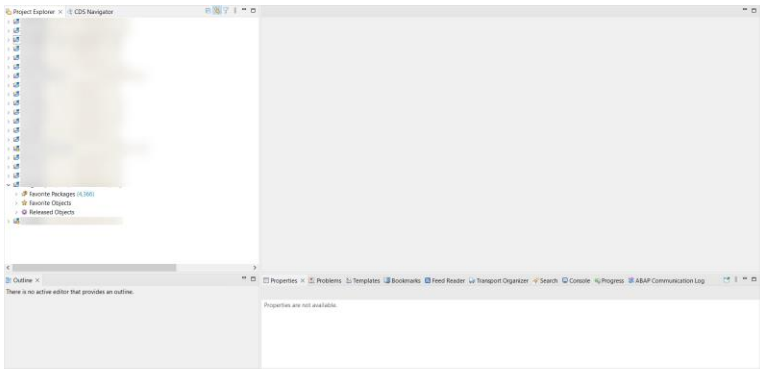
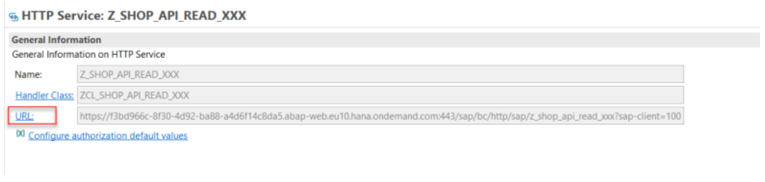
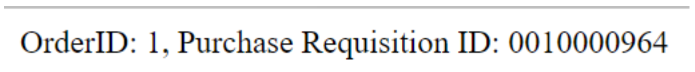
Description	Screenshot
Login to the SAP BTP ABAP Environment instance's Launchpad via your browser. You can find the link in the Cheat Sheet. Login with the user and password from your cheat sheet.	 Home Search in: Apps Read Access Logging Output Management Additional Software Identity and Access Management I18n Services Extensibility System Management Communication Management Communication Arrangements Communication Systems Maintain Communication Users 0 Locked users Maintain Extensions on SAP BTP Display Connectivity Trace Monitor bgRFC Queues Display Communication Scenarios 0 Deprecated in use Application Jobs Application Jobs Application Job Templates Maintain Job Users ABAP Dictionary Repair CDS Views Manage Database Cache Configuration
Open the *Communication Arrangements* Fiori app Create a new communication arrangement, using the previously created custom *Communication Scenario* as your reference	 Communication Arrangements Search in: Apps Standard Editing Status: Arrangement Name: Scenario ID: Communication System: Search Go Adapt Filters Communication Arrangements New Delete Reactivate Arrangement Name Scenario ID Communication System New Communication Arrangement Create Cancel Scenario: * Z_SHOP_SCENARIO_OUTBOUNE Arrangement Name: Z_SHOP_SCENARIO_OUTBOUND_...
As before, press the *New* button next to the *Communication System* field. Choose a System ID and a System Name. After pressing *Create*, you will be forwarded to the configuration page of your new communication system.	 Communication Arrangements Search in: Apps Common Data New Outbound Communication Download Supported Authentication Methods Outbound Services Save Cancel Z_SHOP_SCENARIO_OUTBOUND_XXX Scenario ID: Z_SHOP_SCENARIO_OUTBOUND_XXX Scenario: Exercise 2: Outbound Communication Scenario Draft Last Changed By: Initial Admin Draft Last Changed On: 10/26/2022, 11:01:55 Editing Status: Draft Arrangement Name: Z_SHOP_SCENARIO_OUTBOUND_XXX Communication System: * Own SAP Cloud System: 0M4HTVO API-URL: https://f3bd966c-8f30-4d92-ba88-a4d6f14c8da5.abap.eu10.hana.on... User Name: * Authentication Method: Exercise 2: Outbound Service for HTTP Read Access Service Status: ✓ Active Path: /sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX Application Protocol: HTTP Service Service URL:
Under *Host Name*, add the host name of the S/4HANA Cloud Customizing tenant. You can read the value from the communication arrangement on S/4HANA Cloud side (*API-URL* field).	 Communication System Search in: Apps General Users for Inbound Communication Users for Outbound Communication Communication Arrangements General Data Technical Data General Destination Service OFF Cloud Connector Save Cancel Z_SHOP_API_COMSYS_XXX Z_SHOP_API_COMSYS_XXX Changed By: Initial Admin Changed On: 10/26/2022, 11:03 Editing Status: Draft System ID: * Z_SHOP_API_COMSYS_XXX System Name: * Z_SHOP_API_COMSYS_XXX Notes: Host Name: my400415-api.lab.s4hana.cloud.sap Port: 443 Inbound Only: ☐ UI Host Name: Business System:

Description	Screenshot
Under *Users for Outbound Communication*, add a new technical user via the **+** button. As we are using Basic Authentication, you need to choose the authentication method *User Name and Password*. You will need to specify the same user name and password that you configured on S/4HANA Cloud side. Press *Create*.	 Communication System Z_SHOP_API_COMSYS_XXX Z_SHOP_API_COMSYS_XXX Users for Outbound Communication New Outbound User Authentication Method: * User Name and Password User Name: * Z_SHOP_COMUSER_XXX Password: * •• Communication Arrangements Arrangement Name · Communication Scenario ID · Communication Scenario No data
Save your communication system. You will be redirected back to the original communication arrangement.	 Communication System Z_SHOP_API_COMSYS_XXX Z_SHOP_API_COMSYS_XXX Users for Outbound Communication Authentication Method: User ID and Password User Name / Certificate / Client ID: Z_SHOP_COMUSER_XXX Communication Arrangements Arrangement Name · Communication Scenario ID · Communication Scenario No data
Save your communication arrangement. The previously created communication user will automatically be used. You are now ready to consume your custom API!	 Communication Arrangement Z_SHOP_SCENARIO_OUTBOUND_XXX Scenario ID: Z_SHOP_SCENARIO_OUTBOUND_XXX Scenario: Exercise 2: Outbound Communication Scenario Draft Last Changed By: Initial Admin Draft Last Changed On: 10/26/2022, 11:05:58 Editing Status: Draft Common Data Arrangement Name: Z_SHOP_SCENARIO_OUTBOUND_XXX Communication System: * Z_SHOP_API_COMSYS_XXX Own SAP Cloud System: 0M4HTVO API-URL: https://f3bd966c-8f30-4d92-ba88-a4d6f14c8da5.abap.eu10.hana.on... Outbound Communication User Name: * Z_SHOP_COMUSER_XXX Authentication Method: User ID and Password Outbound Services Exercise 2: Outbound Service for HTTP Read Access Service Status: ✓ Active Application Protocol: HTTP Service Path: /sap/opu/odata/sap/Z_SHOP_API_WEB_V2_XXX Service URL: https://my400415-api.lab.s4hana.cloud.sap:443/sap/o... Draft updated

Step 2.4 – Consume API

As a developer on SAP BTP, retrieve the URL of your HTTP service.

As a business user, call your custom HTTP service.

Description	Screenshot
Login to the SAP BTP ABAP Environment instance via Eclipse	 Project Explorer CDS Navigator Favorite Packages Favorite Objects Released Objects Outline There is no active editor that provides an outline. Properties Problems Templates Bookmarks Feed Reader Transport Organizer Search Console Progress ABAP Communication Log Properties are not available.
Open the HTTP service object. Under *URL*, you will see the endpoint of your service. Copy it into your browser or use the available hyperlink to open it directly from Eclipse.	 HTTP Service: Z_SHOP_API_READ_XXX General Information General Information on HTTP Service Name: Z_SHOP_API_READ_XXX Handler Class: ZCL_SHOP_API_READ_XXX URL: https://f3bd966c-8f30-4d92-ba88-a4d6f14c8da5.abap-web.eu10.hana.ondemand.com:443/sap/bc/http/sap/z_shop_api_read_xxx?sap-client=100 Configure authorization default values
Open the URL in a browser window or click directly on the hyperlink in ABAP Development Tools. Authenticate, if required. Check that the desired data is shown in your browser.	 OrderID: 1, Purchase Requisition ID: 0010000964

Notes

- Your user will automatically have authorization for your custom HTTP service. This is because the Developer business role, which is assigned to your user, automatically grants authorization for all such custom services. To enable its usage for other business users, you will need to perform the necessary Identity and Access management for the service. This is outside the scope of this exercise.
- You can debug the remote call by setting a breakpoint in Eclipse (double-click to the left of the line number marker). We recommend that you set one at the beginning of the `if_http_service_extension~handle_request` method and follow the execution step-by-step.
- In the course of the exercise, we have used some useful ABAP features in our SAP BTP ABAP Environment instance: service consumption models and custom HTTP services. Since S/4HANA Cloud shares the same technology stack, these same features are also available in S/4HANA Cloud Developer Extensibility.

RESOURCES

SAP BTP ABAP Environment – Service Key

```
{
  "binding": {
    "env": "cf",
    "id": "8b09dcaa-d3c3-42f8-b35f-520efc2d818e",
    "type": "oauth",
    "version": "1.0.1.1"
  },
  "catalogs": {
    "abap": {
      "path": "/sap/opu/odata/IWFND/CATALOGSERVICE;v=2",
      "type": "sap_abap_catalog_v1"
    }
  },
  "endpoints": {
    "abap": "https://7e04e291-8515-4d2b-a418-f2868669f491.abap.eu10.hana.ondemand.com"
  },
  "preserve_host_header": true,
  "sap.cloud.service": "com.sap.cloud.abap",
  "systemid": "H01",
  "uaa": {
    "apiurl": "https://api.authentication.eu10.hana.ondemand.com",
    "clientid": "sb-xs-7e04e291-8515-4d2b-a418-f2868669f491!b175432|xsuaa-abapcp-prod-eu10!b4584",
    "clientsecret": "Mo8WI29fWPOFkWYNu+5At3YicSA=",
    "credential-type": "instance-secret",
    "identityzone": "abap-steampunk-v3-2tlbdp49",
    "identityzoneid": "0b5d473f-0308-43e3-945f-a61793af34fb",
    "sburl": "https://internal-xsuaa.authentication.eu10.hana.ondemand.com",
    "subaccountid": "0b5d473f-0308-43e3-945f-a61793af34fb",
    "tenantid": "0b5d473f-0308-43e3-945f-a61793af34fb",
    "tenantmode": "dedicated",
    "uaadomain": "authentication.eu10.hana.ondemand.com",
    "url": "https://abap-steampunk-v3-2tlbdp49.authentication.eu10.hana.ondemand.com",
    "verificationkey": "-----BEGIN PUBLIC KEY-----
\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApwb5GK+uj0E0bn2ImZdJ\nePR7fxWp2kvG9YtmjLP3veddOcYzKc9
J2VlPWQmCyIlXh7DWMLKAByIvNCcz8g0a\nHrRtHHdZkFJEHJt/MwyKIMTirjFjHAcNdIXcQYuLJAfrjNhEIztfv3sWwVCfvPIC\nHrQ
2Nt8vKFuLUIu6pgKuf2e1fWtNgmxpHlae04H1t9GjoqFCJyp9zy4sk2I3PNlG\nmp01ca5RPpnZ3G9IVCTE6xD/Sl5h8CeG2kKma4og
cAQdP2UThQh8TUfQGRe4g2A+\nUunQ5R5oP3DO+PS+4TTTkqSivTXbeLcmBLe7s56JvJHM6oaJwjI3P70PrqFPCo7B\ndQIDAQ
AB\n-----END PUBLIC KEY-----",
    "xsappname": "xs-7e04e291-8515-4d2b-a418-f2868669f491!b175432|xsuaa-abapcp-prod-eu10!b4584",
    "zoneid": "0b5d473f-0308-43e3-945f-a61793af34fb"
  },
  "url": "https://7e04e291-8515-4d2b-a418-f2868669f491.abap.eu10.hana.ondemand.com"
}
```

Sample Code
Please find here some sample code for the HTTP service's handler class:

```
CLASS zcl_shop_api_read_xxx DEFINITION
  PUBLIC
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_http_service_extension .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS zcl_shop_api_read_xxx IMPLEMENTATION.


  METHOD if_http_service_extension~handle_request.

    DATA:
      ls_entity_key    TYPE zonline_shop,
      ls_business_data TYPE zonline_shop,
      lo_http_client   TYPE REF TO if_web_http_client,
      lo_resource      TYPE REF TO /iwbep/if_cp_resource_entity,
      lo_client_proxy  TYPE REF TO /iwbep/if_cp_client_proxy,
      lo_request       TYPE REF TO /iwbep/if_cp_request_read,
      lo_response      TYPE REF TO /iwbep/if_cp_response_read.



    TRY.
        " Create http client
        DATA(lo_destination) = cl_http_destination_provider=>create_by_comm_arrangement(
                                             comm_scenario  = 'Z_SHOP_SCENARIO_OUTBOUND_XXX'
*                                     comm_system_id = '<Comm System Id>'
                                             service_id     = 'Z_SHOP_API_READ_OBS_XXX_REST' ).
        lo_http_client = cl_web_http_client_manager=>create_by_http_destination( lo_destination ).

        lo_client_proxy = cl_web_odata_client_factory=>create_v2_remote_proxy(
          EXPORTING
            iv_service_definition_name = 'Z_SHOP_API_SCM_XXX'
            io_http_client             = lo_http_client
            iv_relative_service_root   = '' ).


        " Set entity key
        ls_entity_key = VALUE #(
                  order_uuid  = '5B3F46742F341EEDA5C9E32C19BEDA01' ).

        " Navigate to the resource
        lo_resource = lo_client_proxy->create_resource_for_entity_set( 'ONLINE_SHOP' )->navigate_with_key(
ls_entity_key ).

        " Execute the request and retrieve the business data
        lo_response = lo_resource->create_request_for_read( )->execute( ).
        lo_response->get_business_data( IMPORTING es_business_data = ls_business_data ).

        DATA: result TYPE string.
        result = | OrderID: { ls_business_data-Order_Id },  Purchase Requisition ID: { ls_business_data-
Purchasereqn  } |.
        response->set_text( result ).


      CATCH /iwbep/cx_cp_remote INTO DATA(lx_remote).
        " Handle remote Exception
        " It contains details about the problems of your http(s) connection
        response->set_text( | Remote Error: { lx_remote->get_longtext(  ) } | ).
      CATCH /iwbep/cx_gateway INTO DATA(lx_gateway).
        " Handle Exception
        response->set_text( | Gateway Error: { lx_gateway->get_longtext(  ) } | ).
      CATCH cx_http_dest_provider_error INTO DATA(lx_destination).
        "handle exception
        response->set_text( | Destination Error: { lx_destination->get_longtext(  ) } | ).
      CATCH cx_web_http_client_error INTO DATA(lx_http).
        "handle exception
        response->set_text( | HTTP Error: { lx_http->get_longtext(  ) } | ).
    ENDTRY.
  ENDMETHOD.
ENDCLASS.
```