

SAP NetWeaver Gateway – Service Development

How to develop a Gateway Service using code based implementation

Applies to:
SAP NetWeaver Gateway 2.0 SP6 and
SAP NetWeaver 7.40 SP2

Author:
Andre Fischer, SAP AG, <http://scn.sap.com/people/andre.fischer>

TABLE OF CONTENTS

HOW TO GUIDE OBJECTIVES.....	3
BUSINESS EXAMPLE	3
OVERVIEW OF TASKS	4
Task 1: Creating a Data Model with entity type Product and the entity set Products	4
Task 2: Generate runtime objects and register and activate the service	4
Task 3: Perform Service Implementation for GET_ENTITYSET	4
Task 4: Perform Service Implementation for GET_ENTITY	4
HOW TO SECTION	5
Task 1: Creating a Data Model with entity Product and the entity set ProductCollection	5
Step: Create a new project in the Service Builder (transaction SEGW)	5
Step: Import DDIC Structure to create the entity - Product	5
Step: Create an entity set for the previously created entity	8
Task 2: Generate runtime objects and register and activate the service	9
Step: Generate runtime objects	9
Step: Test the new OData service (metadata only)	12
Task 3: Service Implementation for GET_ENTITYSET	15
Step: Implement the data provider class	15
Task 4 Service Implementation for GET_ENTITY	18
Step: Implement the method GET_ENTITY operation of the entityset Products	19
Step: Test GET_ENTITY method	21

HOW TO GUIDE OBJECTIVES

In this How to Guide, you will learn how to quickly create an OData service with SAP NetWeaver Gateway. Unlike the usual “Hello World” sample we will leverage demo data from the Enterprise Procurement Model which is part of every SAP NetWeaver ABAP server as of 7.02.

In this How to Guide we will introduce the central development tool the Service Builder and will explain the basics of implementing services using the OData channel programming model.

In this How to Guide, you will learn how to create a data model based on a DDIC structure and implement the appropriate methods in the data provider extension class.

After completing this lesson, you will be able to:

- Create a data model with one entity set and one entity type where the entity type is based on a DDIC structure
- Implement the GET_ENTITYSET method in the data provider extension class
- Implement the GET_ENTITY method in the data provider extension class

BUSINESS EXAMPLE

You want to build an application that shows a list of products.

OVERVIEW OF TASKS

Task 1: Creating a Data Model with entity type Product and the entity set Products

1. Create a new project ZGW_PRODUCT.
2. Import DDIC Structure *BAPI_EPM_PRODUCT_HEADER* to create the entity type – *Product*
3. Create an entity set *Products* which is based on the entity type *Product* you have created before.

Task 2: Generate runtime objects and register and activate the service

1. Generate the runtime objects of your project
2. Register the service in the Gateway hub system (which is in this case the local system since we are using Gateway in an embedded deployment scenario)
3. Test the service using the Gateway Client.

Task 3: Perform Service Implementation for GET_ENTITYSET

1. Implement the method *PRODUCTS_GET_ENTITYSET* in the extension class of the data provider class
2. We will use the function module '*BAPI_EPM_PRODUCT_GET_LIST*' to retrieve the product data
3. Please note that the parameter *ET_ENTITYSET* that contains the data returned by the *GET_ENTITYSET* method has the same structure as the structure *BAPI_EPM_PRODUCT_HEADER* that has been used to generate the entity type *Product*.

Task 4: Perform Service Implementation for GET_ENTITY

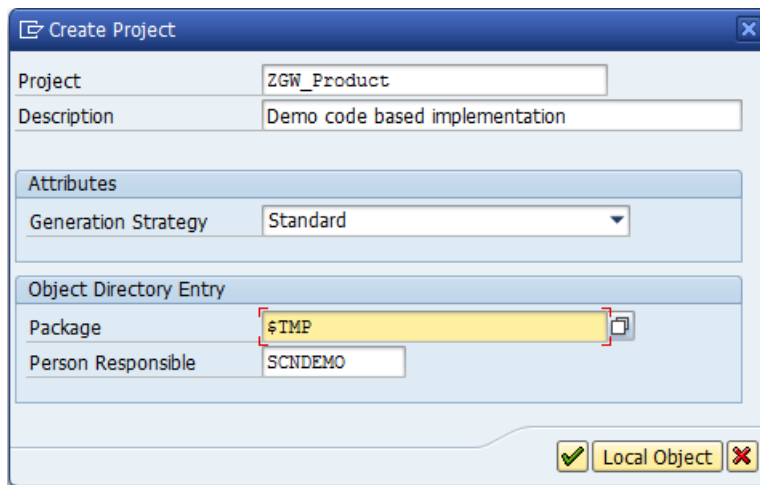
1. Implement the method *PRODUCTS_GET_ENTITY* in the extension class of the data provider class
2. We will use the function module '*BAPI_EPM_PRODUCT_GET_DETAIL*' to retrieve the product details
3. Please note that the parameter *ER_ENTITY* that contains the data returned by the *GET_ENTITY* method has the same structure as the structure *BAPI_EPM_PRODUCT_HEADER* that has been used to generate the entity type *Product*

HOW TO SECTION

Task 1: Creating a Data Model with entity Product and the entity set Products

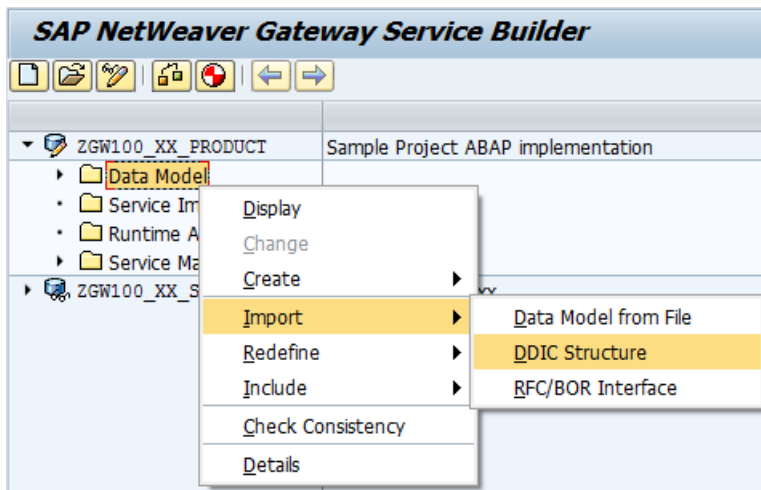
Step: Create a new project in the Service Builder (transaction SEGW)

1. Create a new project **ZGW_PRODUCT**



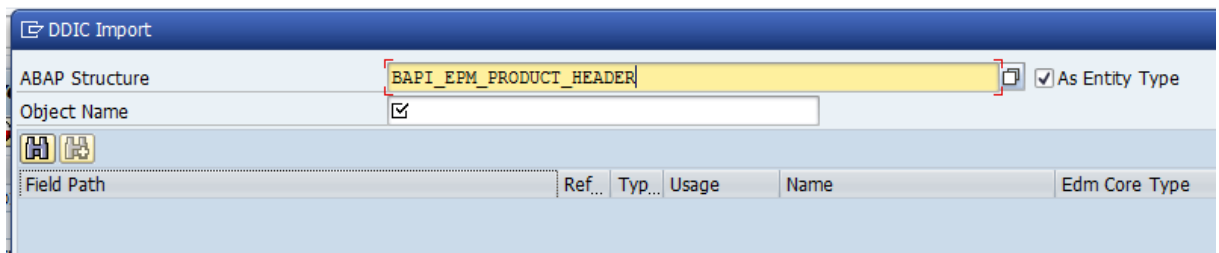
Step: Import DDIC Structure to create the entity - Product

1. Right-click *Data Model* and choose *Import* → *DDIC Structure*



2. Enter the following value in the wizard and then press Enter:

ABAP Structure: **BAPI_EPM_PRODUCT_HEADER**



and press ENTER.

Result:

The wizard will automatically fill the field ObjectName with the value *BapiEpmProductHeader* and create an entity set based on the DDIC structure.

3. Change the default values such that

- a. the property *PRODUCT_ID* becomes a key field
- b. the name of the entity type is *Product* rather than *BapiEpmProductHeader*

DDIC Import

ABAP Structure: BAPI_EPM_PRODUCT_HEADER As Entity Type

Object Name: Product

Field Path	Ref...	Typ...	Usage	Name	Edm Core Type
PRODUCT_ID	<input type="checkbox"/>	<input type="checkbox"/>	Key	ProductID	Edm.String
TYPE_CODE	<input type="checkbox"/>	<input type="checkbox"/>	Property	TypeCode	Edm.String
CATEGORY	<input type="checkbox"/>	<input type="checkbox"/>	Property	Category	Edm.String
NAME	<input type="checkbox"/>	<input type="checkbox"/>	Property	Name	Edm.String
DESCRIPTION	<input type="checkbox"/>	<input type="checkbox"/>	Property	Description	Edm.String
SUPPLIER_ID	<input type="checkbox"/>	<input type="checkbox"/>	Property	SupplierID	Edm.String
SUPPLIER_NAME	<input type="checkbox"/>	<input type="checkbox"/>	Property	SupplierName	Edm.String
TAX_TARIF_CODE	<input type="checkbox"/>	<input type="checkbox"/>	Property	TaxTarifCode	Edm.Byte
MEASURE_UNIT	<input type="checkbox"/>	<input type="checkbox"/>	Property	MeasureUnit	Edm.String
WEIGHT_MEASURE	<input type="checkbox"/>	<input type="checkbox"/>	Property	WeightMeasure	Edm.Decimal
WEIGHT_UNIT	<input type="checkbox"/>	<input type="checkbox"/>	Property	WeightUnit	Edm.String
PRICE	<input type="checkbox"/>	<input type="checkbox"/>	Property	Price	Edm.Decimal
CURRENCY_CODE	<input type="checkbox"/>	<input type="checkbox"/>	Property	CurrencyCode	Edm.String
WIDTH	<input type="checkbox"/>	<input type="checkbox"/>	Property	Width	Edm.Decimal
DEPTH	<input type="checkbox"/>	<input type="checkbox"/>	Property	Depth	Edm.Decimal
HEIGHT	<input type="checkbox"/>	<input type="checkbox"/>	Property	Height	Edm.Decimal
DIM_UNIT	<input type="checkbox"/>	<input type="checkbox"/>	Property	DimUnit	Edm.String
PRODUCT_PIC_URL	<input type="checkbox"/>	<input type="checkbox"/>	Property	ProductPicUrl	Edm.String


4. As a result the values should like follows.

DDIC Import

ABAP Structure: BAPI_EPM_PRODUCT_HEADER

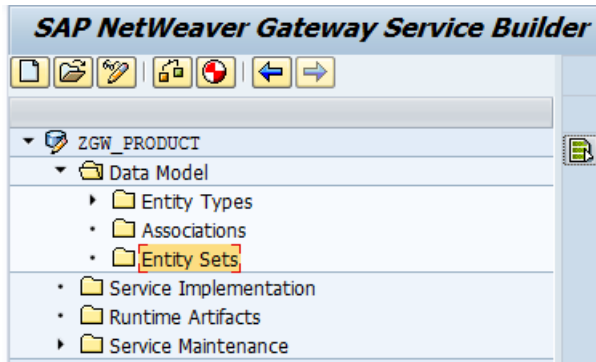
Object Name: Product

Field Path	Ref...	Typ...	Usage	Name
PRODUCT_ID	<input type="checkbox"/>	<input type="checkbox"/>	Key	ProductID

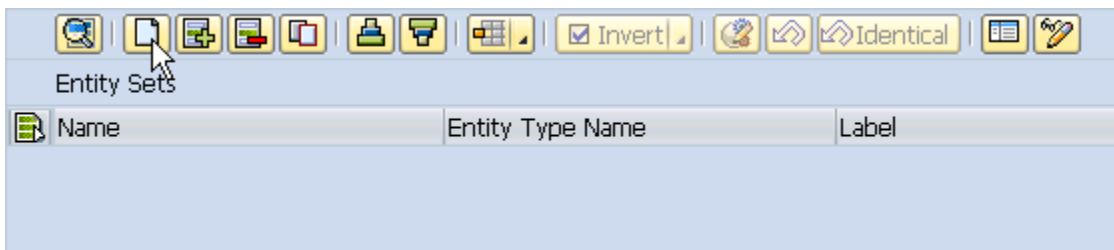
5. Press continue .
6. Press Save.

Step: Create an entity set for the previously created entity

1. Expand the node *Data Model* and double-click *Entity Sets*:



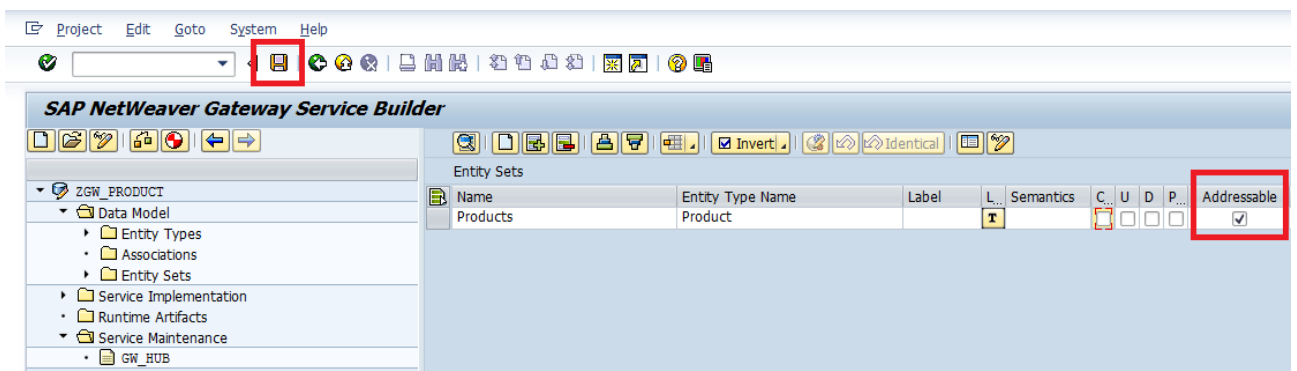
2. Click the *Create* button to a new line to the table:



3. Enter the following values:

Name	Entity Type Name	Adressable
Products	Product	X

4. Choose *Save*:



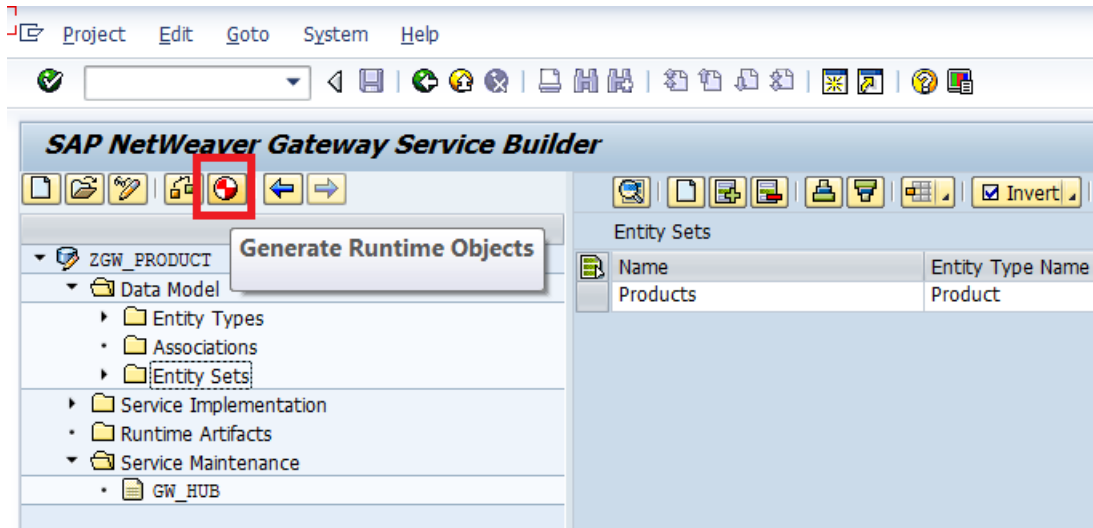
Caution:

Please make sure that the *Addressable* flag is marked because otherwise GWPA will refuse to generate a consumer application.

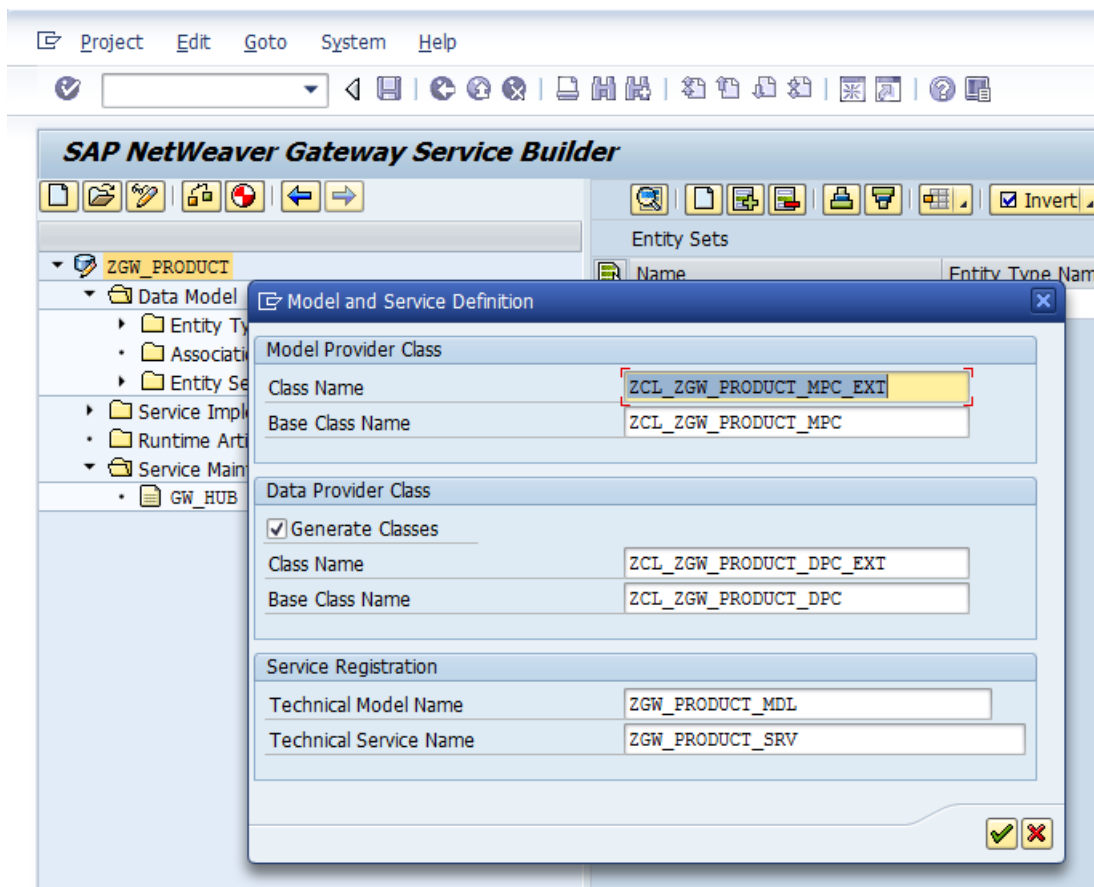
Task 2: Generate runtime objects and register and activate the service

Step: Generate runtime objects

1. Choose the *Generate* pushbutton:



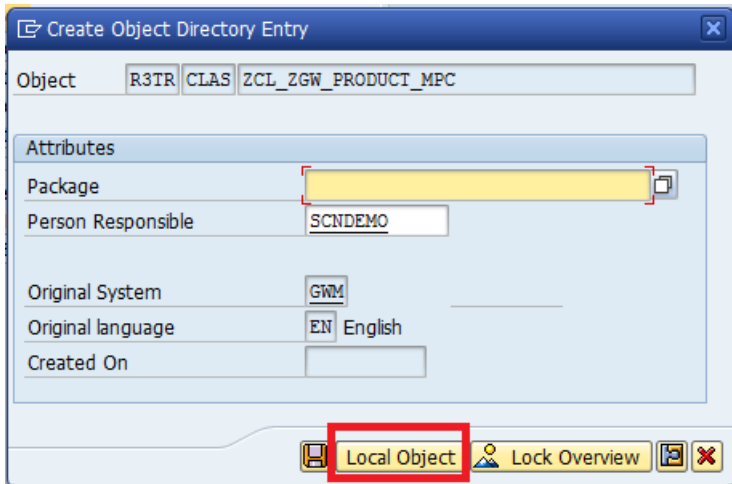
2. Leave the default values and choose *Enter*.



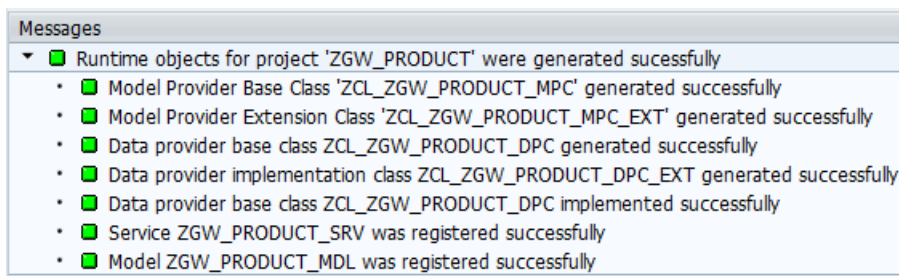
Please note the Technical Service Name *ZGW_PRODUCT_SRV* is equal to the External Service

Name required to consume this service later on.

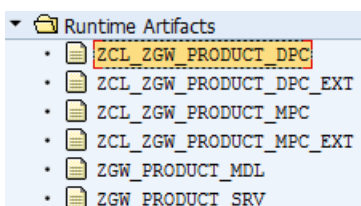
3. Choose *Local Object*:



4. Verify that the runtime objects have been generated successfully:

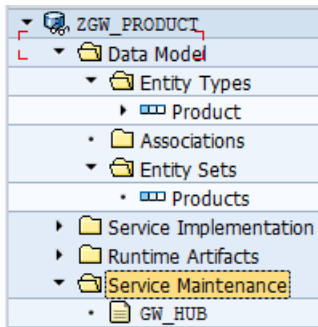


The following objects have been generated:

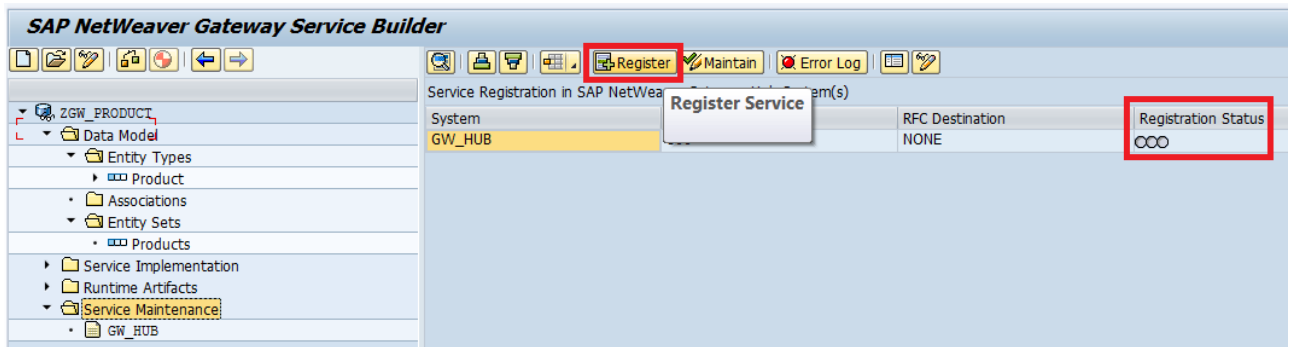


Step: Register and activate the service

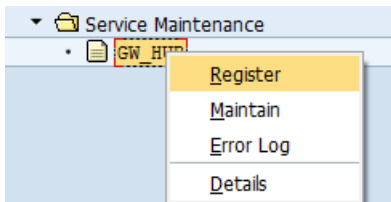
1. Double-click *Service Maintenance*:



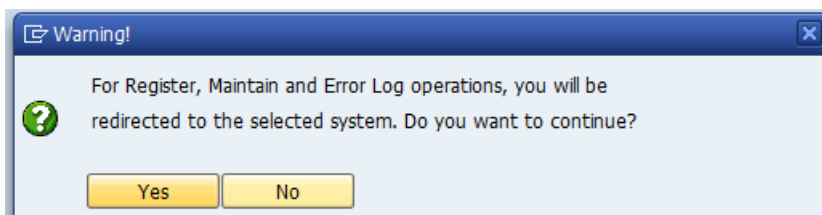
2. Select system GW_HUB and choose the *Register* button:



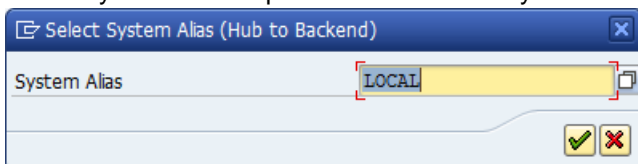
Alternatively you can click on the entry GW_HUB on the left hand side, right click and select Register from the context menu.



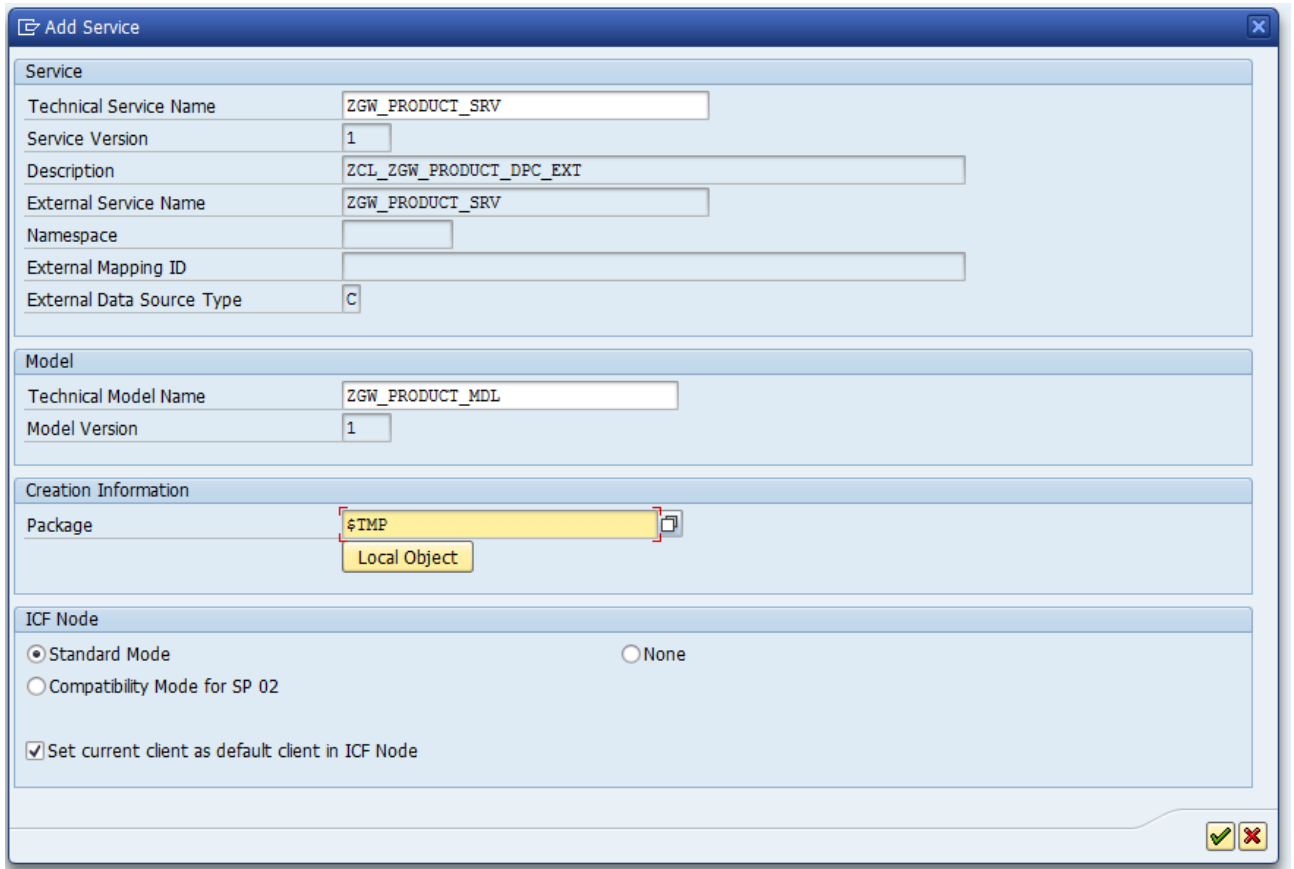
3. Confirm the warning message displayed in the popup:



4. (Optional) If you have defined multiple system alias entries in the Gateway hub system that point back to your backend press F4 to select a system alias and confirm the *Select System Alias* popup:

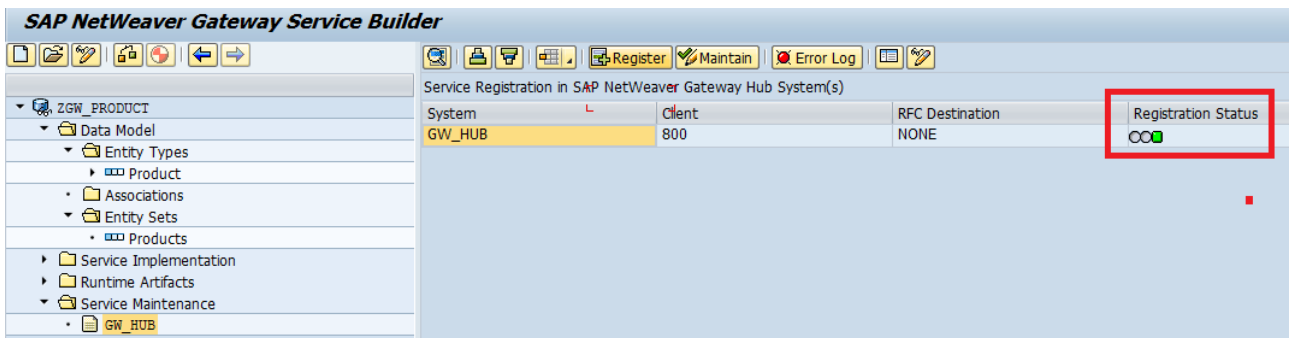


- In the *Add Service* dialogue leave the default values and enter *\$tmp* as the package and choose *Enter* or press the button *Local Object*:



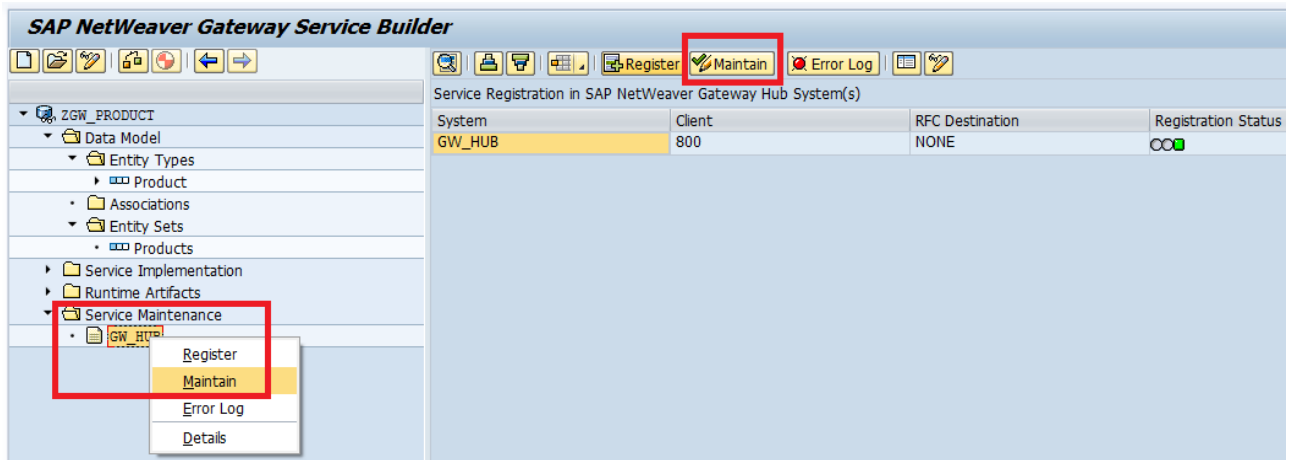
The External Service Name ZGW_PRODUCT_SRV is defaulted with the Technical Service Name from the previous step.

- Verify that the service has been registered and activated successfully:

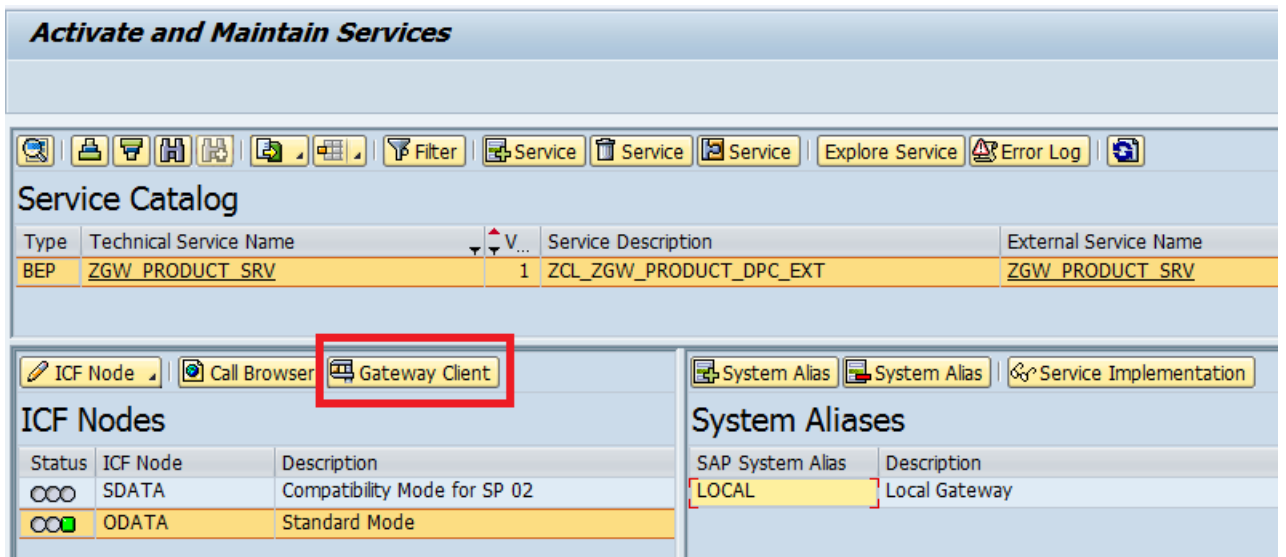


Step: Test the new OData service (metadata only)

- Right click the system in which you have registered the service and select *Maintain*. Alternatively you can select the system in the mass maintenance view and select the *Maintain* button.



2. This will start the *Activate and Maintain Services* transaction `/IWFND_MAINT_SERVICE` where the service you have just registered is selected.



3. Press the button `Gateway Client`. This will open in a new window start transaction `/IWFND/GW_CLIENT` using the the URI `/sap/opu/odata/sap/ZGW_PRODUCT_SRV/?$format=xml` and choose *Execute*. This will show the service document of the newly created service.

SAP NetWeaver Gateway Client

Execute Select Save Maintain Service Service Implementation

HTTP Method: GET POST PUT PATCH DELETE HEAD OPTIONS

Request URI: /sap/opu/odata/sap/ZGW_PRODUCT_SRV/?\$format=xml

Test Group: Test Case

HTTP Request

Header Name	Value
Header Name	Value

HTTP Response

Header Name	Value
~status_code	200
~status_reason	OK
~server_protocol	HTTP/1.0
set-cookie	sap-XSRF_GWM_800=KogRkV2BwnjKcoW5M0zAg%3d%3d20130618073328fMk...
content-type	application/xml
content-length	833
x-csrf-token	KogRkV2BwnjKcoW5M0zAg==
cache-control	no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0

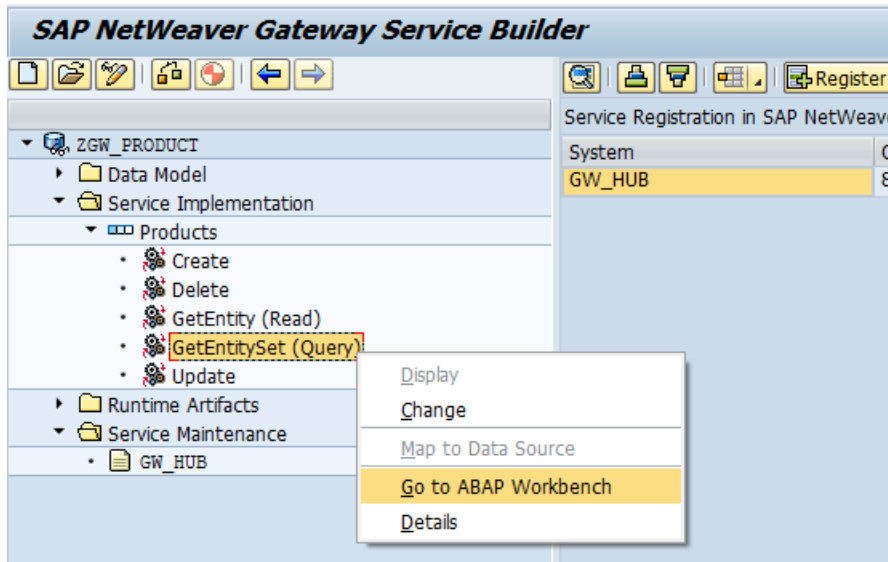
```
<?xml version="1.0" encoding="utf-8" ?>
- <app:service xml:lang="en"
  xml:base="http://vegtwy1mst.wdf.sap.corp:50000/sap/opu/odata/sap/ZGW_PRODUCT_SRV/"
  xmlns:app="http://www.w3.org/2007/app" xmlns:atom="http://www.w3.org/2005/Atom"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
  xmlns:sap="http://www.sap.com/Protocols/SAPData">
- <app:workspace>
  <atom:title type="text">Data</atom:title>
- <app:collection sap:pageable="false" sap:content-version="1" href="Products">
  <atom:title type="text">Products</atom:title>
  <sap:member-title>Product</sap:member-title>
  </app:collection>
</app:workspace>
<atom:link rel="self"
  href="http://vegtwy1mst.wdf.sap.corp:50000/sap/opu/odata/sap/ZGW_PRODUCT_SRV/" />
<atom:link rel="latest-version"
  href="http://vegtwy1mst.wdf.sap.corp:50000/sap/opu/odata/sap/ZGW_PRODUCT_SRV/" />
</app:service>
```

Please note that ZGW_PRODUCT_SRV is the External Service Name that was registered before.

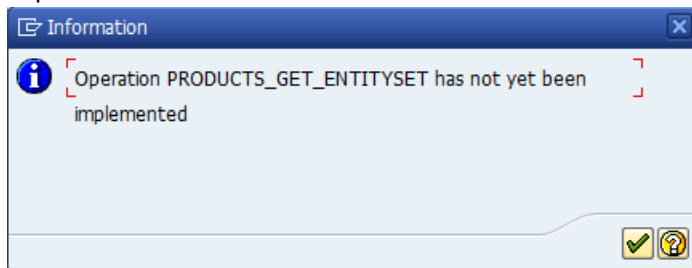
Task 3: Service Implementation for GET_ENTITYSET

Step: Implement the data provider class

1. In the Service Builder go back using the green arrow and expand the node *Service Implementation*, then the Entity Set *Products* and right click on the entry *GetEntitySet (Query)*.



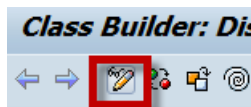
2. Confirm the warning that the method PRODUCTS_GET_ENTITYSET has not yet been implemented.



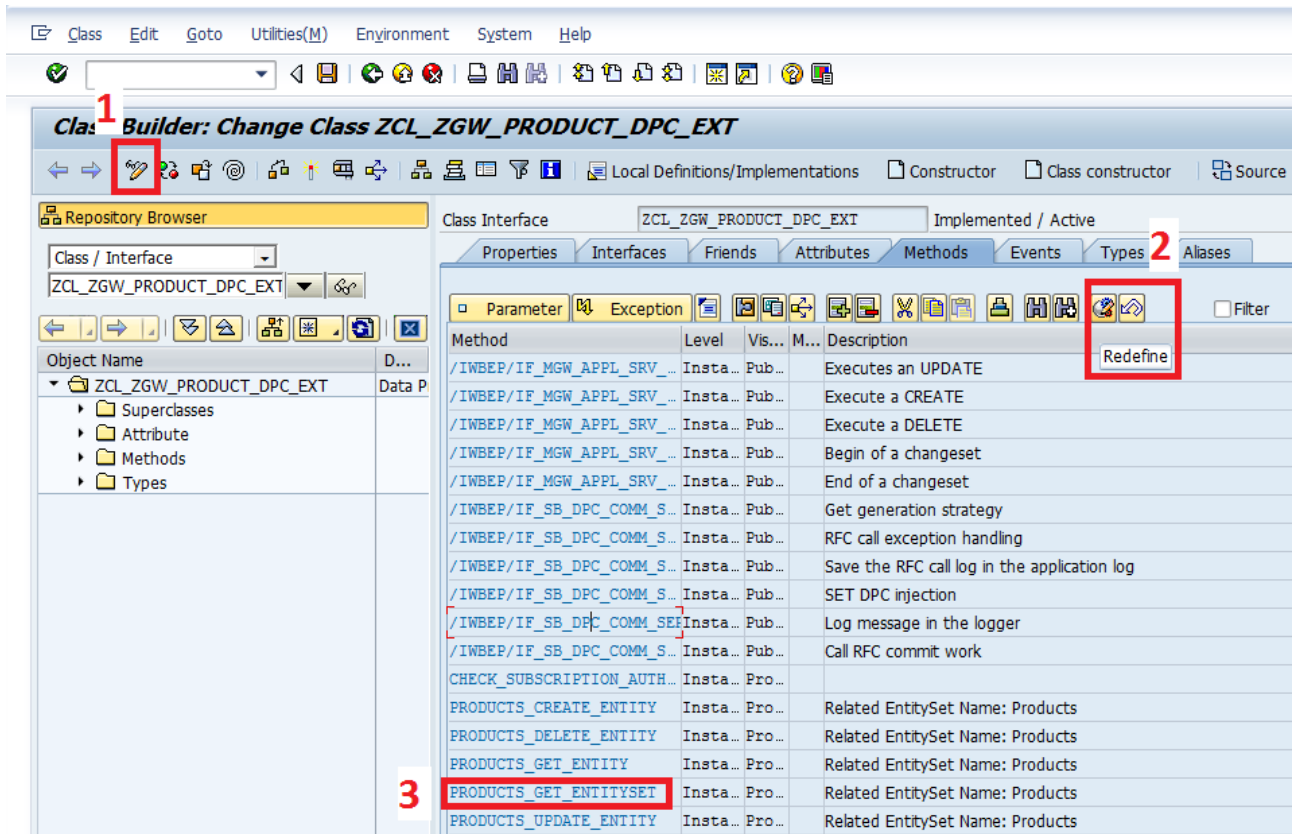
This will open the Class Builder.

Alternatively expand the node *Runtime Artifacts* and double-click on *ZCL_ZGW_PRODUCT_DPC_EXT*

3. Activate the Change mode



4. Redefine method PRODUCTS_GET_ENTITYSET.



5. Implement Method PRODUCTCOLLECTIO_GET_ENTITYSET
 - a. You may copy the code from ZCL_ZGW_PRODUCT_DPC_EXT-> PRODUCTS_GET_ENTITYSET or
 - b. Perform the following steps:
 - i. Choose *Pattern*
 - ii. Select *Call Function* and enter the value *BAPI_EPM_PRODUCT_GET_LIST*
 - iii. Assign *et_entityset* to *headerdata*

```

Method | PRODUCTS_GET_ENTIIYSET | Inactive
-----|-----|-----
1 | method PRODUCTS_GET_ENTITYSET .
2 |
3 | CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_LIST'
4 | * EXPORTING
5 | *   MAX_ROWS           =
6 | TABLES
7 |   HEADERDATA          = et_entityset
8 | * SELPARAMPRODUCTID   =
9 | * SELPARAMSUPPLIERNAME =
10 | * SELPARAMCATEGORIES  =
11 | * RETURN              =
12 |
13 |
14 | endmethod.
    
```

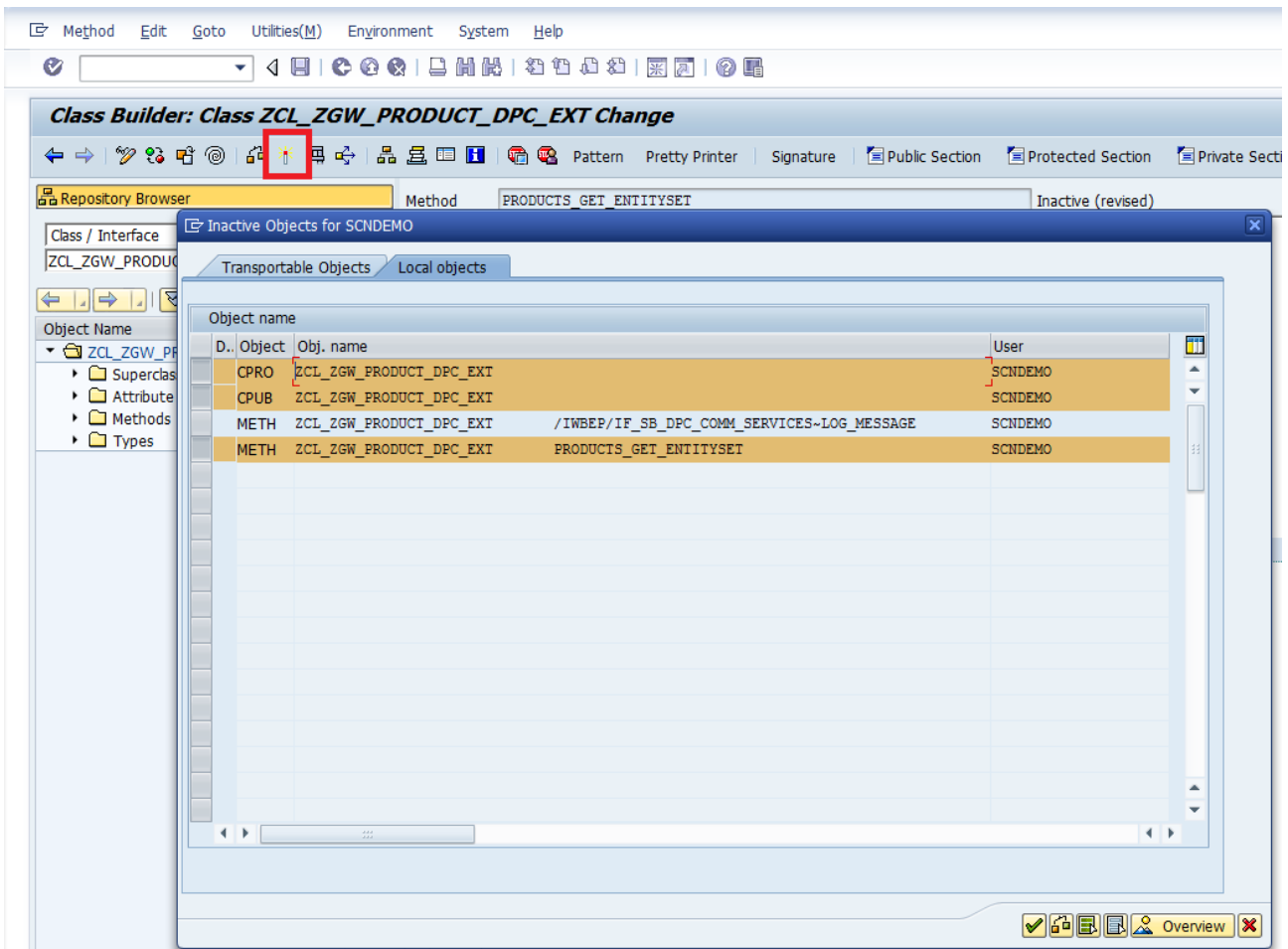
- c. Or take the code from here using Cut & Paste

d. Activate the coding

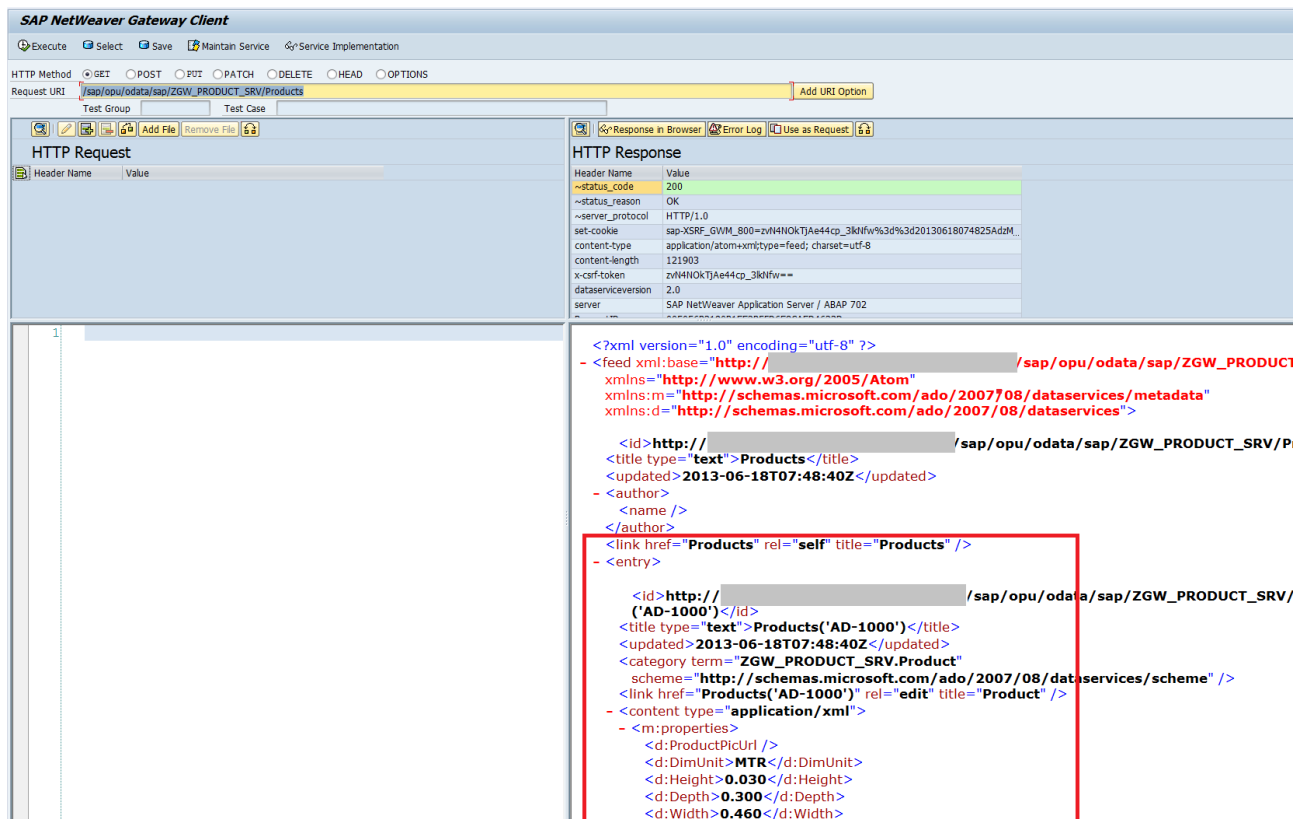
```
method PRODUCTS_GET_ENTITYSET.

CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_LIST'
* EXPORTING
*   MAX_ROWS           =
TABLES
  HEADERDATA          = et_entityset
*   SELPARAMPRODUCTID =
*   SELPARAMSUPPLIERNAMES =
*   SELPARAMCATEGORIES =
*   RETURN             =
.

endmethod.
```



Start the Gateway Client (Transaction /IWFND/GW_CLIENT) in a separate window to run the service.
 Provide the following URI to get the Product Collection:
/sap/opu/odata/sap/ZGW_PRODUCT_SRV/Products



Task 4 Service Implementation for GET_ENTITY

Prerequisites

- You have performed Task 1 to 3 and you have performed the following steps
 - o Project creation
 - o You have created a model with the entity set *Products* which is based on an entity type *Product* that has been created using import of the DDIC structure *BAPI_EPM_PRODUCT_HEADER*
 - o Implementation of the GET_ENTITYSET method

Objectives

In the following you will learn how to implement the GET_ENTITY method. For this we have to learn how to retrieve the key(s)

If in the Gateway Client following GET statement is issued

```
/sap/opu/odata/sap/ZGW_PRODUCT_SRV/Products('AD-1000')
```

the framework takes care that the method PRODUCTS_GET_ENTITY of the data provider extension class ZCL_ZGW_PRODUCT_DPC_EXT is called.

In order to provide the caller with the requested content the key value 'AD-1000' shall be retrieved using the import parameter IO_TECH_REQUEST_CONTEXT.

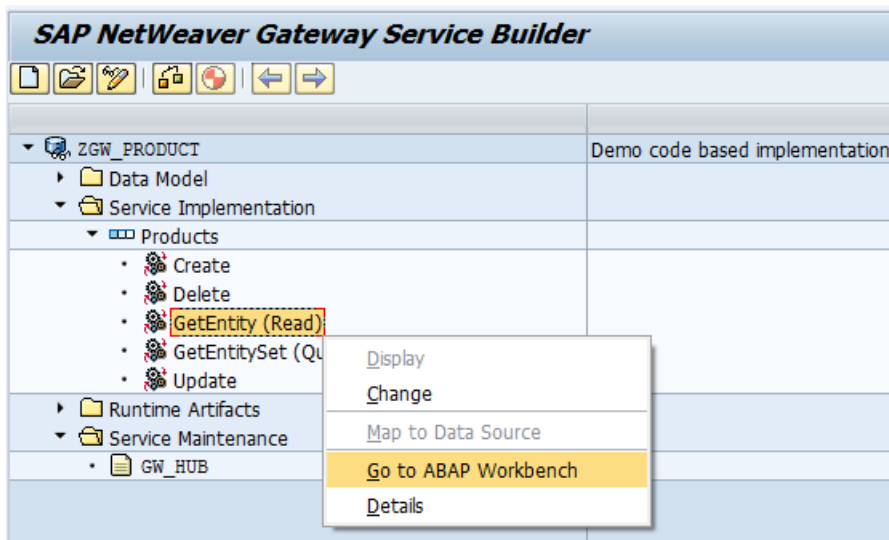
This import parameter “speaks” the technical (internal) names and hence is not influenced by any redefine function which could result that the external name of the key field can potentially be changed. The internal names - unless explicitly set by the model provider - are derived from the "original" name.

We can set an external breakpoint in the PRODUCTS_GET_ENTITY method to check that the key value ‘AD-1000’ can be retrieved in the ABAP code as a key-value pair from the internal table. This table would contain several entries in case the entity set has more than one key field.

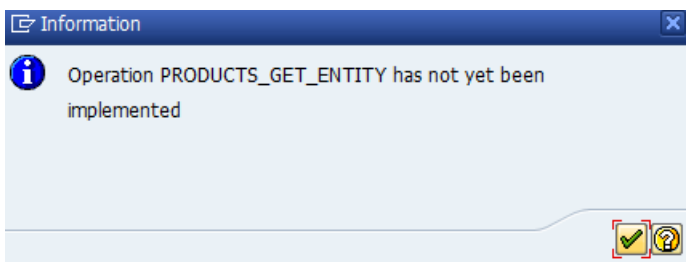
The parameter ER_ENTITY which is used to return the data has the same type as the data dictionary object that has been imported namely *BAPI_EPM_PRODUCT_HEADER*.

Step: Implement the method GET_ENTITY operation of the entityset Products

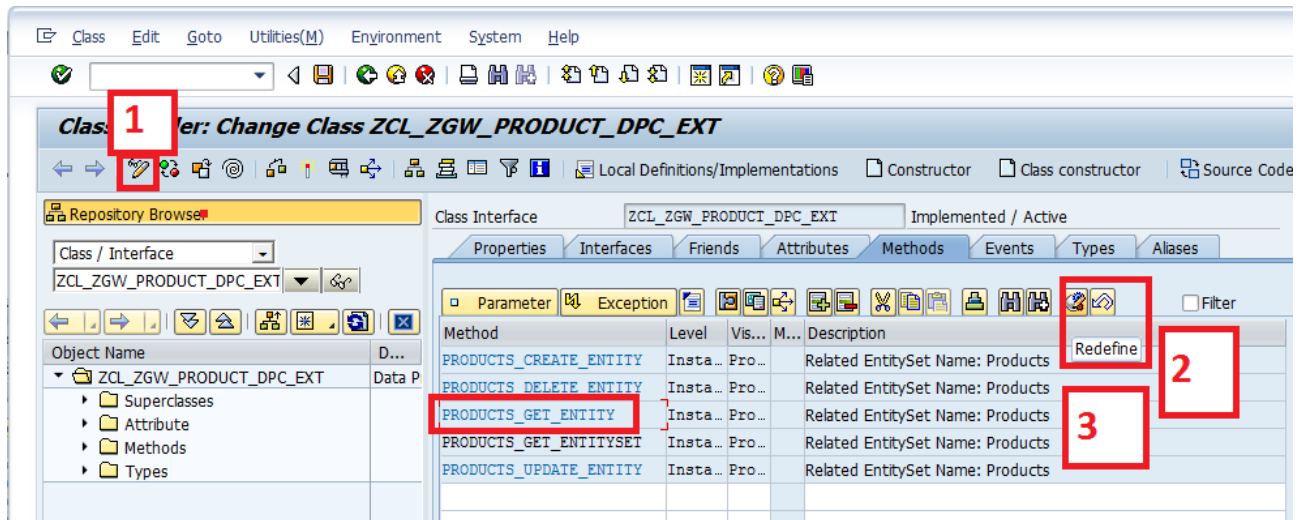
1. Open the Service Builder project ZGW_PRODUCT that you have created in the previous exercise.
2. Expand the node *Service Implementation*, drill down to the entity set *Products* and expand the same. Right click on the *GetEntity* method and select *Go to ABAP Workbench*



3. Confirm the message “Operation PRODUCTS_GET_ENTITY has not yet been implemented”



4. Scroll down to the method PRODUCTS_GET_ENTITY (1), switch to the edit mode (2) and choose redefine (3)



5. You can retrieve the sample code from the following code snippet.

```
method PRODUCTS_GET_ENTITY.

DATA:   lt_keys TYPE /IWBEF/T_MGW_TECH_PAIRS,
        ls_key  TYPE /IWBEF/S_MGW_TECH_PAIR,
        ls_product_id type BAPI_EPM_PRODUCT_ID.

lt_keys = IO_TECH_REQUEST_CONTEXT->GET_KEYS( ).

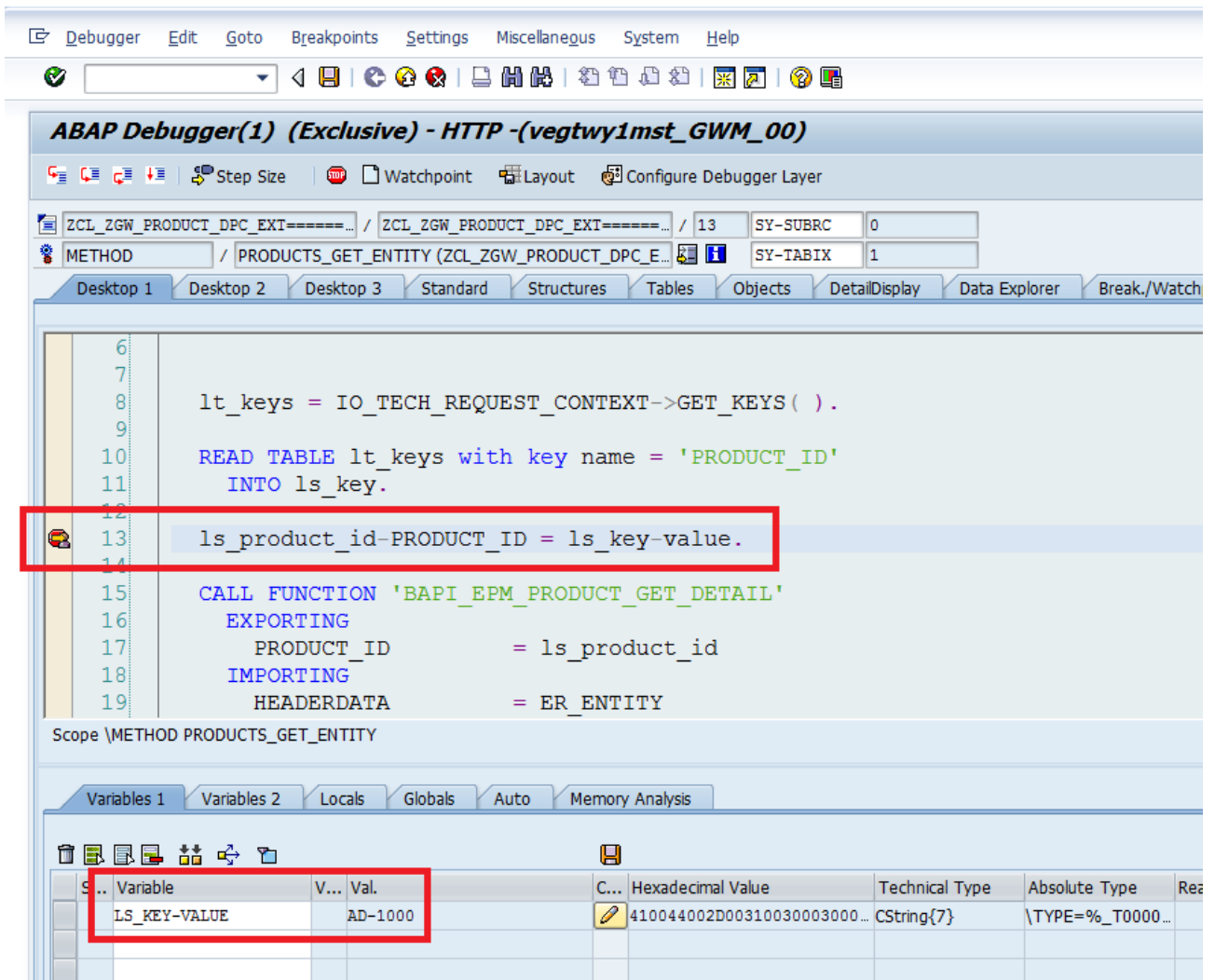
READ TABLE lt_keys with key name = 'PRODUCT_ID'
  INTO ls_key.

ls_product_id-PRODUCT_ID = ls_key-value.

CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_DETAIL'
  EXPORTING
    PRODUCT_ID           = ls_product_id
  IMPORTING
    HEADERDATA          = ER_ENTITY
  * TABLES
  * RETURN              = lt_return
  .

endmethod.
```

6. (Optional) You can check how the data is retrieved in the ABAP code if you set an external breakpoint



Step: Test GET_ENTITY method

1. Check the implementation by using the following request URI in the Gateway Client **/sap/opu/odata/sap/ZGW_PRODUCT_SRV/Products('AD-1000')**

The screenshot displays the SAP NetWeaver Gateway Client interface. The top section shows the HTTP Method set to GET and the Request URI as `/sap/opu/odata/sap/ZGW_PRODUCT_SRV/Products('AD-1000')`. The HTTP Response section shows a status code of 200 and a detailed XML body. The XML body contains an entry for the product 'AD-1000' with a title, updated timestamp, category, and a link to the product details.

```

<?xml version="1.0" encoding="utf-8" ?>
- <entry
  xml:base="http://          /sap/c
  xmlns="http://www.w3.org/2005/Atom"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/
  xmlns:d="http://schemas.microsoft.com/ado/2007/08/c
    <id>http://          /sap/opu/
    ('AD-1000')</id>
    <title type="text">Products('AD-1000')</title>
    <updated>2013-06-18T09:01:47Z</updated>
    <category term="ZGW_PRODUCT_SRV.Product"
      scheme="http://schemas.microsoft.com/ado/2007/08/
    <link href="Products('AD-1000')" rel="edit" title="Product"
  - <content type="application/xml">
  - <m:properties>
    <d:ProductPicUrl />
    <d:ProductPicUrl MTP />
  
```

You are done 😊

© 2013 SAP AG. All rights reserved.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

