

ODATA

What is OData and REST in SAP Netweaver Gateway

- SAP Netweaver Gateway provides REST based and open interface that implements simple access to SAP systems via OData protocol.
- With SAP Netweaver Gateway any one can i.e. any user interface(UI) can access the data through OData protocol.
- Is this the only one that can communicate to other systems in SAP portfolio? the answer is NO. We have SAP PI system to exchange information between different systems and especially between a company and 3rd parties.
- But the main difference between SAP PI and SAP Netweaver Gateway is in their usage.
- SAP Netweaver PI is to communicate the information between system-to-system, but SAP Netweaver Gateway is system-to-user.

What is OData?

- OData is REST based protocol to access and update the data. OData is build on Atom Publishing Protocol(AtomPub), XML and JSON. This make the protocol easy to understand and use.
- In short OData is an Online Database Connectivity(ODBC) for the web. It offers a simple and uniform way of sharing the data on protocol level which in turn enables broad integration across different products and platforms.

What is REST?

- OData is REST-inspired technology for reading, writing, and modifying information on the Web (not just SAP). **REST** = **RE**presentational **S**tate **T**ransfer. *REST is an **architectural style** that uses simple and lightweight mechanism for inter-machine communication.* It is an alternative to the RPC (Remote Procedure Calls) and Web Services. REST is **Resource-based**, unlike RPC or SOAP which are **Action-based**.
- REST services are called as REST services because the Services are really working with **Resources** instead of **Operations**. Any communication between client and services are using **URI** (Unified Resource Identifier) over HTTP protocol using HTTP method. The URI is really the representation of the Resources (like POHeader, POItem, Customer, Vendor etc). Also, in RESTful service, once you identified the Resource, you will be working with a uniform interface, because it uses HTTP methods (GET, PUT, POST and DELETE) to work with the resource. *So, the client does not need to know what the exact operation name defined in the service contract to call that method.* GET method is used whenever we need to get the representation of an existing resource. POST is used to add new resource into the system. PUT is to modify the existing resource and DELETE is to remove the resource from the system. *No matter what is the Service in whatever Platform, GET, PUT, POST, DELETE remains the same.*

Structure of OData Service

It have mainly two parts

1. Service document
2. Service metadata document.

Service document

- Service documents consists of all list of resources URI's that can be accessible. ZSL_EPM_DEMO is the OData service which we will be going to build in future tutorials. Basically this service will retrieve sales order data.
- Lets look at what information does this service documents holds. Service document is accessible through the URI.

SAP NetWeaver Gateway Client

Execute Select Maintain Service Service Implementation EntitySets

HTTP Method ☒ GET ☐ POST ☐ PUT ☐ PATCH ☐ MERGE ☐ DELETE ☐ Reuse HTTP Connection (e.g. necessary for Soft State)

Request URI Add URI Option

Protocol ☒ HTTP ☐ HTTPS Test Group Test Case

```
<?xml version="1.0" encoding="utf-8" ?>
- <app:service xml:lang="en" xml:base="http://localhost:8080/sap/opu/odata/SAP/ZSL_EPM_DEMO_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:creatable="false" sap:updatable="false" sap:deletable="false" sap:pageable="false" sap:content-version="1" href="SalesOrderSet">
    <atom:title type="text">SalesOrderSet</atom:title>
    <sap:member-title>SalesOrder</sap:member-title>
  </app:collection>
  - <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:pageable="false" sap:content-version="1" href="OrderItemsSet">
    <atom:title type="text">OrderItemsSet</atom:title>
    <sap:member-title>OrderItems</sap:member-title>
  </app:collection>
</app:workspace>
<atom:link rel="self" href="http://localhost:8080/sap/opu/odata/SAP/ZSL_EPM_DEMO_SRV/" />
<atom:link rel="latest-version" href="http://localhost:8080/sap/opu/odata/SAP/ZSL_EPM_DEMO_SRV/" />
```

Service metadata document

- Service metadata documents contains meta data of all elements in the service. You can see the metadata of a service by simple adding “\$metadata” to the service URI.

The screenshot shows a web client interface with the following fields:

- HTTP Method: ☒ GET ☐ POST ☐ PUT ☐ PATCH ☐ MERGE ☐ DELETE
- Request URI: `/sap/opu/odata/SAP/ZSL_EPM_DEMO_SRV/$metadata` (highlighted with an orange box)
- Protocol: ☒ HTTP ☐ HTTPS
- Test Group:
- Test Case:
- Reuse HTTP Connection (e.g. necessary for Soft State): ☐
- Add URI Option:

The response is an XML document representing the service metadata. A portion of the XML is highlighted with an orange box:

```
<?xml version="1.0" encoding="utf-8" ?>
- <edmx:Edmx Version="1.0" xmlns:edmx="http://schemas.microsoft.com/ado/2007/06/edmx"
  xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:sap="http://www.sap.com/Protocols/SAPData">
- <edmx:DataServices m:DataServiceVersion="2.0">
- <Schema Namespace="ZSL_EPM_DEMO_SRV" xml:lang="en" sap:schema-version="0000" xmlns="http://schemas.microsoft.com/ado/2008/09/edm">
  - <EntityType Name="SalesOrder" sap:content-version="1">
    - <Key>
      <PropertyRef Name="SoId" />
    </Key>
    <Property Name="SoId" Type="Edm.String" Nullable="false" MaxLength="10" sap:label="Sa. Ord. ID" sap:creatable="false" sap:updatable="false" sap:sortable="false"
      sap:filterable="false" />
    <Property Name="CurrencyCode" Type="Edm.String" Nullable="false" MaxLength="5" sap:label="Currency" sap:creatable="false" sap:updatable="false"
      sap:sortable="false" sap:filterable="false" sap:semantics="currency-code" />
    <Property Name="GrossAmount" Type="Edm.Decimal" Nullable="false" Precision="16" Scale="3" sap:unit="CurrencyCode" sap:label="Gross Amt."
      sap:creatable="false" sap:updatable="false" sap:sortable="false" sap:filterable="false" />
    <Property Name="NetAmount" Type="Edm.Decimal" Nullable="false" Precision="16" Scale="3" sap:unit="CurrencyCode" sap:label="Net Amt." sap:creatable="false"
      sap:updatable="false" sap:sortable="false" sap:filterable="false" />
    <Property Name="TaxAmount" Type="Edm.Decimal" Nullable="false" Precision="16" Scale="3" sap:unit="CurrencyCode" sap:label="Tax Amt." sap:creatable="false"
      sap:updatable="false" sap:sortable="false" sap:filterable="false" />
  </EntityType>
```

Advantages of an ODATA

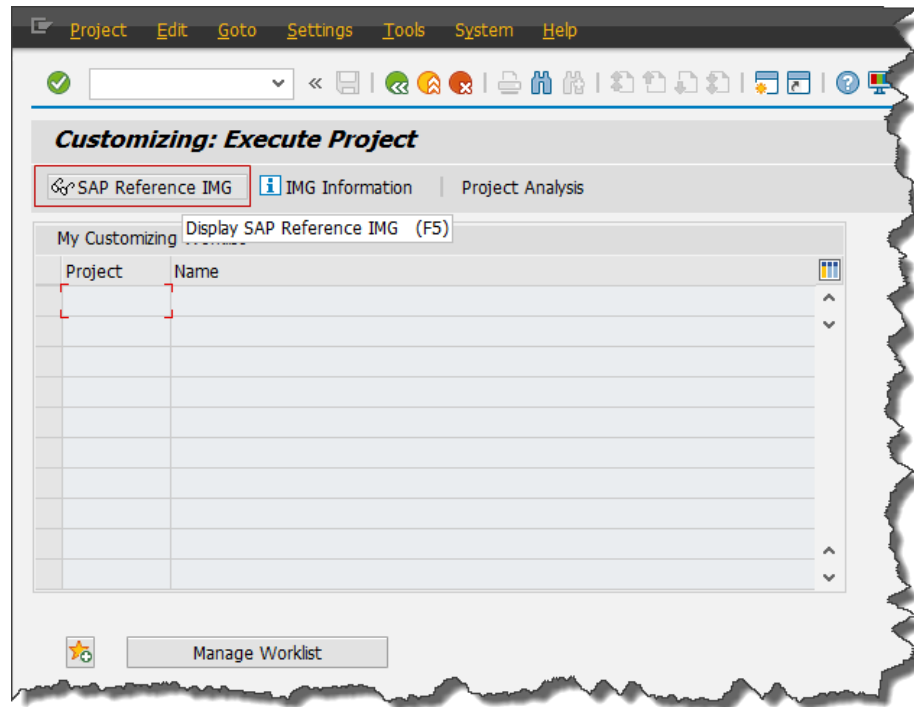
- Using SAP OData provides us following advantages:
- It helps to obtain human readable results i.e. you can use your browser to see the output data
- It is very easy and relatively fast to access data
- It uses all the standards of web protocols i.e. GET, PUT, POST, DELETE, and QUERY
- It uses Stateless Applications: It means Server does not save any data of Client (e.g. UI5 Application) and treats every OData call as a new call
- It receives data in form of related pieces of information, one leading to another: It is an interaction pattern known as “alert-analyse-act”, “view-inspect-act”, or “explore & act”. According to this pattern not all data are loaded together, and a user analyses a data and reaches its required information after navigation. In this way the data loads quickly and correctly.

Different elements in SAP OData service

- **Entity Type:** Entity is like work area which holds empty or one record data. As we have different fields in work area here also we have different fields and we call them as Properties. Each Entity should have at least one key field.
- **Entity Set:** Entity Set is a collection of same entity types. It is like internal table which holds n records of same type. For example list of sales orders is a Entity Set.
- **Property:** It represents a primitive data type element. It is like a single field in a work area or single column in a table. one or more properties are used to create an Entity Types.
- **Association:** It defines the relation between different entity types. For example if we have two entity types one for Sales Order header and other for Sales Order Item we can build the association between these two entity types with cardinality.
- **Navigation property:** Entity Types include one or more navigation properties. It is specific type which acts like a link to the other Entity types based on cardinality provided in the Association property. To create the navigation property for an entity type we need first define the association between those entity types.

How to Activate SAP Netweaver Gateway

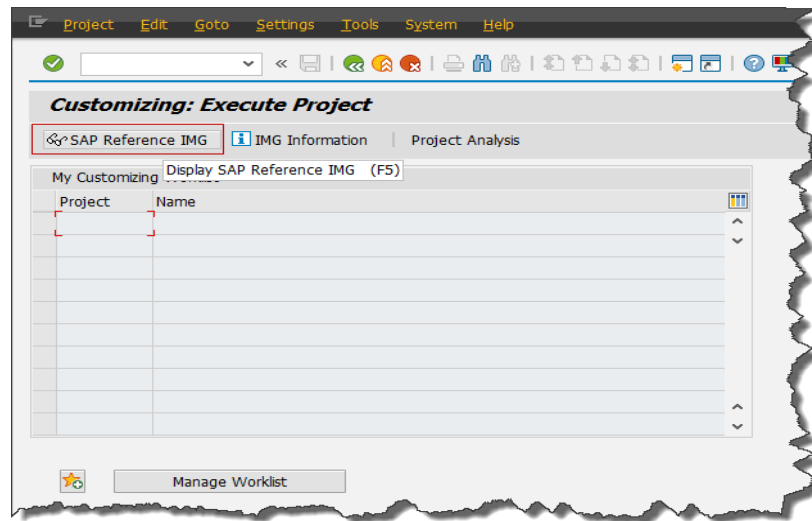
1. Go to SPRO >> SAP Reference IMG



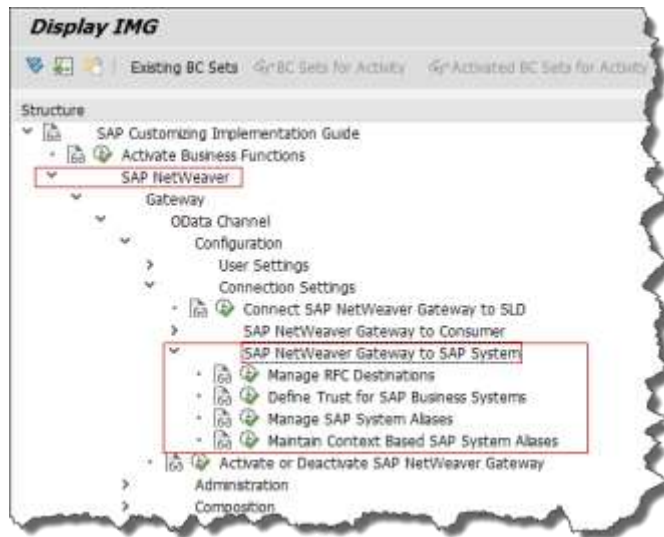
2. Expand *SAP Netweaver >> Gateway >> ODATA Channel >> Configuration >> Activate or Deactivate SAP Netweaver Gateway*
3. Click on Activate or Deactivate button to activate or deactivate SAP Netweaver Gateway.

How to Connect SAP Gateway to Backend Systems

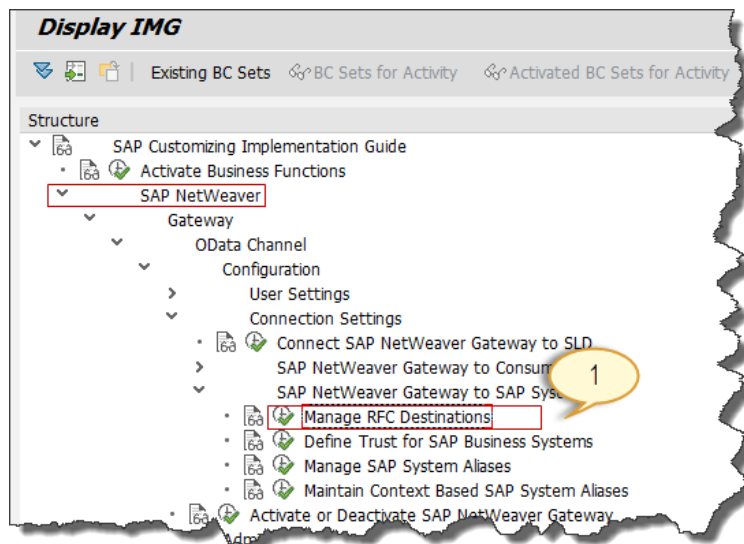
1. Go to SPRO >> SAP Reference IMG



2. Expand SAP Netweaver >> Gateway >> ODATA Channel >> Configuration >> Connection Settings >> SAP Netweaver Gateway to SAP System.



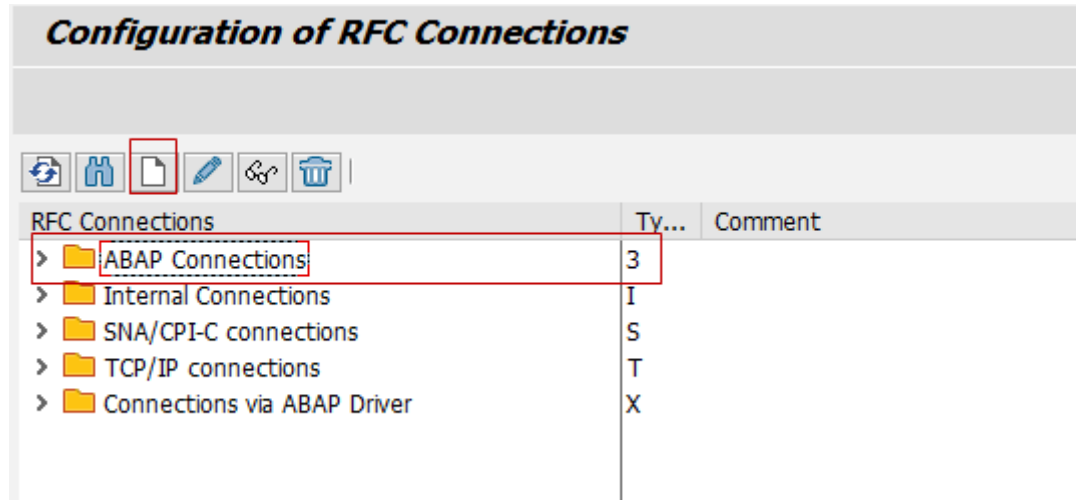
3. Execute “Mange RFC Destination”



4. Create a new ABAP RFC Connection to SAP Back-end Systems (ERP, SRM, CRM etc.).

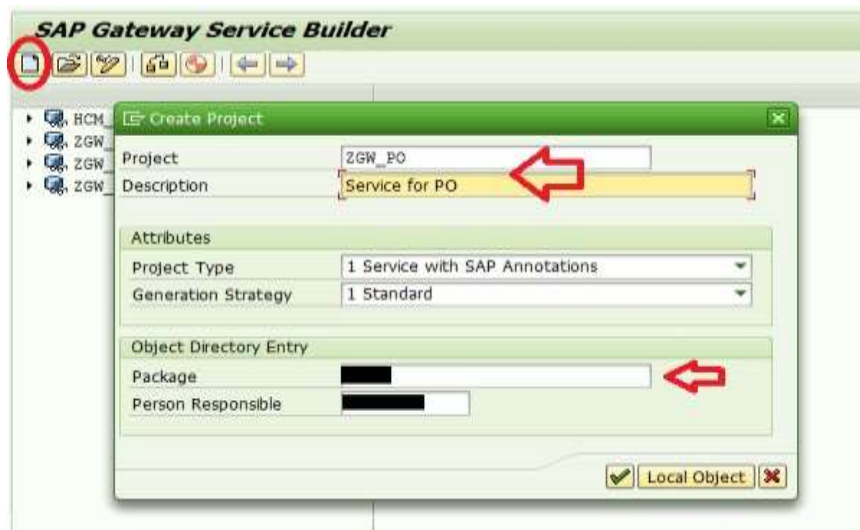
If you choose Hub Deployment, create a RFC connection to connect to SAP ERP system.

If you choose Embedded Deployment, create a RFC connection to itself.



Simple Demo on ODATA Creation

- Go to t-code **SEGW** (SAP Gateway Service Builder). Hit the Create Icon and provide the name of the Project, description and package (or local) and save it.



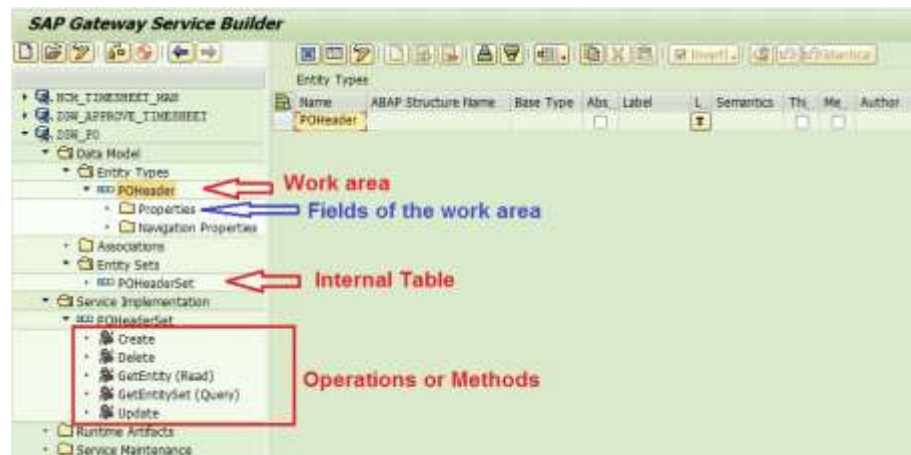
- The project gets created with four folders, namely **Data Model**, **Service Implementation**, **Runtime Artifacts** and **Service Maintenance**. Please take note that Data Model further has three sub-folders viz **Entity Types**, **Associations** and **Entity Sets**. All the folders are empty by default.



- Entity Type is our very own structure (or a work area (holds just one row)). And you guessed it right, Entity Set is an internal table (holds more than one entity/rows).
- Let us create our first structure



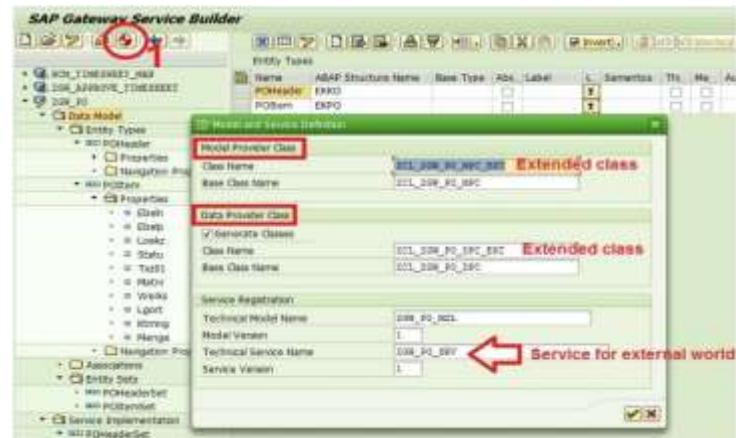
- Right click on Entity Types folder and select “Create”, provide the name you like and do not forget to tick the checkbox “Create Related Entity Set”. For our example, POHeader is the structure(work area) while POHeaderSet is our internal table.



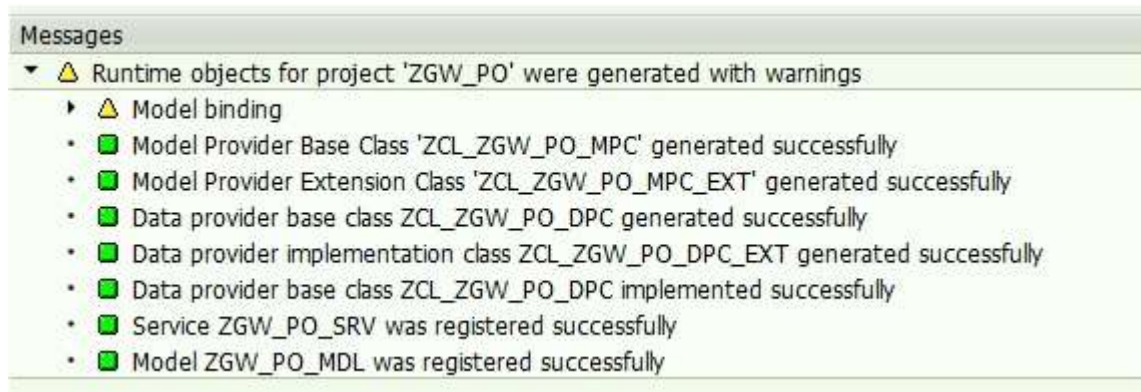
Check the Service Implementation folder has POHeaderSet Operations auto generated. *These are ABAP Methods which would be triggered when the relevant endpoints would be called.*

Implement/Register the Service

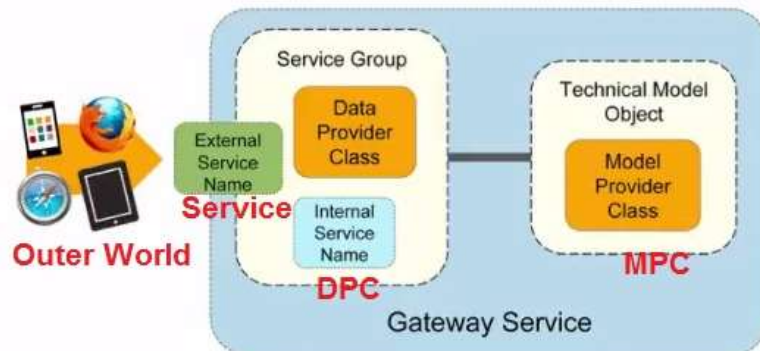
Let us go ahead and generate and register our service



- Hit the **Generate** icon and hit ok. Provide the package and transport number or save it as local. You should get the success message as shown below.



- Please note the **Technical Service Name is the actual Service which the external system needs to call**. Two classes, **Model Provider Class (MPC)** and **Data Provider Class (DPC)** are also generated along with Base and Extended Class.



- **DPC and MPC are connected by Configuration (Not Coding)**

- Model Provider Class inherits from /IWBEP/CL_MGW_ABS_MODEL and Data Provider Class inherits from /IWBEP/CL_MGW_ABS_DATA. The below image shows the relationship between the generated classes and their superclasses (parents).

SAP Gateway Service Builder

Runtime Artifacts

Name	Generated Artifact Type	Program ID	Object Type
ZCL_ZGW_PO_DPC	Data Provider Base Class	R3TR	CLAS
ZCL_ZGW_PO_DPC_EXT	Data Provider Extension Class	R3TR	CLAS
ZCL_ZGW_PO_MPC	Model Provider Base Class	R3TR	CLAS
ZCL_ZGW_PO_MPC_EXT	Model Provider Extension Class	R3TR	CLAS
ZGW_PO_MD	Registered Model	R3TR	IVMO
ZGW_PO_SRV	Registered Service	R3TR	IVSV

Class/Interface: ZCL_ZGW_PO_DPC Implemented / Active

Superclass: /IWBEP/CL_MGW_PUSH_ABS_DATA

Description: Data Provider Base Class

Inst. Generation: 3 Abstract

Class/Interface: /IWBEP/CL_MGW_PUSH_ABS_DATA Implemented / Active

Superclass: /IWBEP/CL_MGW_ABS_DATA

Description: Abstract Push Data Provider

Inst. Generation: 3 Abstract

Class/Interface: ZCL_ZGW_PO_MPC Implemented / Active

Superclass: /IWBEP/CL_MGW_PUSH_ABS_MODEL

Description: ZCL_ZGW_PO_MPC

Inst. Generation: 2 Public

Class/Interface: /IWBEP/CL_MGW_PUSH_ABS_MODEL Implemented / Active

Superclass: /IWBEP/CL_MGW_ABS_MODEL

Description: Abstract Push Model Provider

Inst. Generation: 2 Public

- Model Provider Class - This is used to define model. you can use the method Define to create entity, properties etc using code based implementation. you rarely use MPC extension class.
- Data Provider Class - used to code your CRUDQ methods as well as function import methods. you write all your logic in redefined methods of DPC extension class.

Add Service to Service Catalog (Register the Service to Gateway Hub)

- We have implemented the Service, now we need to add the service to the ***Service Catalog***.

The screenshot displays the SAP Gateway Service Builder interface. On the left, a tree view shows the project structure, with 'Service Maintenance' and 'GW_HUB' highlighted. The main area shows 'Service Registration in SAP Gateway Hub System(s)' with a table containing one entry for 'GW_HUB' with client '100' and RFC Destination 'NONE'. The 'Registration Status' column shows three small circles, with a red arrow pointing to it. Below the table, a 'Messages' section lists several success messages, including 'Runtime objects for project ... were generated successfully' and 'Service Z_EPM_PRODUCTS_SRV was registered successfully'. A red arrow points to the 'Service Maintenance' folder in the left tree.

System	Client	RPC Destination	Registration Status
GW_HUB	100	NONE	○○○

Messages:

- Runtime objects for project ... were generated successfully
- Model Provider Base Class ... generated successfully
- Model Provider Extension Class ... generated successfully
- Data provider base class ... generated successfully
- Data provider implementation class ... generated successfully
- Data provider base class ... implemented successfully
- Service Z_EPM_PRODUCTS_SRV was registered successfully
- Mode ... was registered successfully

- Go to your **Gateway Hub (Front-end system)** and execute t-code /n/IWFND/MAINT_SERVICE

Activate and Maintain Services

Filter Add Service Delete Service Service Details Load Metadata Error Log

Request Statistics Refresh Catalog OAuth Soft State Processing Mode

Service Catalog

Type	Technical Service Name	V...	Service Description	External Service Name	Resp.	0
BEP	C_DRAFTLIFECYCLEADMINDATA_CDS	1	Draft Lifecycle Administrative Data	C_DRAFTLIFECYCLEADMINDATA_CDS		
	/IWFND/SG_MED_CATALOG	1	Catalog Service	CATALOGSERVICE	/IWFND/	
	/IWFND/SG_MED_CATALOG	2	Catalog Service Version 2	CATALOGSERVICE	/IWFND/	
BEP	DAAG_DTG_SRV	1	Odata for DTG app	DAAG_DTG_SRV		

ICF Node Call Browser SAP Gateway Client

ICF Nodes

Status	ICF Node	Session Time-out	Soft State	Description
CO	ODATA	00:00:00		Standard Mode

Add System Alias Remove System Alias Outamping

System Aliases

SAP System Alias	Description	Defa
------------------	-------------	------

- Add the service to the Service Catalog, only then is our service available for the outer world to access.

Add Selected Services

Get Services 2

Filter:

System Alias:

Technical Service Name: Version:

External Service Name: External Mapping ID:

Select Backend Services

Type	Technical Service Name	Version	Service Description	External Service Name
BEP	ZGW_PO_SRV	1	Service for PO	ZGW_PO_SRV

3. Click here

Add Service

Service

Technical Service Name:

Service Version:

Description:

External Service Name:

Namespace:

External Mapping ID:

External Data Source Type:

Model

Technical Model Name: Model Version:

Creation Information

Package Assignment:

ICF Node

☒ Standard Mode ☐ None

☒ Set Current Client as Default Client in ICF Node

OAuth enablement:

☐ Enable OAuth for Service

- Hit the Add Service button, provide your backend system alias and external service name (our case ZGW_PO_SRV). You will get the service you created in the backend. Click on it and it would show the Service (technical/external) name along with the Technical name of the Model (ZGW_PO_MDL) and hit save. Go back to the Service Catalog screen.

Activate and Maintain Services

Filter Add Service Delete Service Service Details Load Metadata Error Log Request Statistics

Refresh Catalog OAuth Soft State Processing Mode

Service Catalog

Type	Technical Service Name	V	Service Description	External Service Name	Map.	OAuth	Soft State	Processing Mode
BEP	REPM_SALESORDERDETAILSO_CDS	1	REPM: Sales Order Details Query View	REPM_SALESORDERDETAILSO_CDS			Not Supported	Routing-based
BEP	REPM_SALESORDERPRODUCTQUERY_CDS	1	REPM: Sales Order Product Query View	REPM_SALESORDERPRODUCTQUERY_CDS			Not Supported	Routing-based
BEP	ZS_EPM_SADL_GW_DEV_SCEN_BO_S	1	EPM: SADL-based GW-Service Developer Scenario	S_EPM_SADL_GW_DEV_SCEN_BO_SRV			Not Supported	Routing-based
BEP	SADL_V_EXP_QUERY_CDS	1	SADL Generated Service	SADL_V_EXP_QUERY_CDS				Routing-based
BEP	SADL_V_SALESORDER_BO_CDS	1	SADL Generated Service	SADL_V_SALESORDER_BO_CDS			Not Supported	Routing-based
BEP	SEPM_HANA_EXT_PAL_ODATA_SRV	1	SEPM_HANA_EXT_PAL_ODATA	SEPM_HANA_EXT_PAL_ODATA_SRV			Not Supported	Routing-based
BEP	/IWFND/SUBSCRIPTIONMANAGEMENT	2	MOC enabled Subscription Management Service	SUBSCRIPTIONMANAGEMENT	/IWFND/			Routing-based
BEP	ZTRANSPORT	1	UI2: Transport Service	TRANSPORT	/UI2/			Routing-based
	/IWFND/USAGEEXTRACTOR	1	Metering Usage Extractor	USAGEEXTRACTOR	/IWFND/			Routing-based
	/IWFND/SG_USER_SERVICE	1	Information Worker - User Service	USERSERVICE	/IWFND/			Routing-based
BEP	ZGW_APPROVE_TIMESHEET_SRV	1	Approve Timesheet Extension	ZGW_APPROVE_TIMESHEET_SRV			Not Supported	Routing-based
BEP	ZGW_PO_SRV	1	Service for PO	ZGW_PO_SRV			Not Supported	Routing-based
BEP	ZGW_PURCHASE_SRV	1	Purchase Order App	ZGW_PURCHASE_SRV			Not Supported	Routing-based
BEP	ZHCM_TIMESHEET_APPROVE_SRV_1	1	Approve Timesheet Extension	ZHCM_TIMESHEET_APPROVE_SRV			Not Supported	Routing-based

ICF Node Call Browse SAP Gateway Client 3

ICF Nodes

Status	ICF Node	Session Time-out	Soft State	Description
OK	ODATA	00:00:00		Standard Mode

System Aliases 2 This shows up

SAP System Alias	Description	Default	M
UD1_130	UD1 Backend		

- Let us test it using **SAP Gateway Client**. Or use t-code /IWFND/GW_CLIENT (remember this t-code as well). You can also test by Call Browser option. For now, we will use SAP Gateway Client option. status code is 200 i.e success.

The screenshot displays the SAP Gateway Client interface. At the top, there are tabs for 'Execute', 'Select', 'Service Administration', 'Service Implementation', 'Entity Sets', and 'Add URI Option'. Below these, the 'HTTP Method' is set to 'GET', and the 'Request URI' is '/sap/opu/odata/sap/ZGW_PO_SRV/?format=xml'. The 'Protocol' is 'HTTP'. The 'HTTP Response - Processing Time = 137 ms' is shown. A table of headers is displayed, with 'status_code' highlighted in yellow and circled in green, showing a value of '200'. The 'status_reason' is 'OK'. Below the headers, the XML response is shown, starting with '<?xml version="1.0" encoding="utf-8" ?>'. The XML structure includes a service definition with various namespaces and a collection of POHeaderSet and POItemSet.

Header Name	Value
status_code	200
status_reason	OK
sap-process-time	ms/chub=,cp=,st=,MedCacheTab=SHM,codeployed=,softstate=
last-modified	Thu, 29 Dec 2016 23:07:55 GMT

```
<?xml version="1.0" encoding="utf-8" ?>
- <app:service xmlns:lang="en"
  xmlns:base="http://txaixegd01[redacted]/sap/opu/odata/sap/ZGW_PO_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:creatable="false" sap:updatable="false" sap:deletable="false"
  sap:pageable="false" sap:content-version="1" href="POHeaderSet">
  <atom:title type="text">POHeaderSet</atom:title>
  <sap:member title="POHeader"></sap:member>
</app:collection>
- <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false"
  sap:pageable="false" sap:content-version="1" href="POItemSet">
  <atom:title type="text">POItemSet</atom:title>
  <sap:member title="POItem"></sap:member>
</app:collection>
</app:workspace>
<atom:link rel="self"
  href="http://txaixegd01[redacted]/sap/opu/odata/sap/ZGW_PO_SRV/" />
<atom:link rel="latest-version"
  href="http://txaixegd01[redacted]/sap/opu/odata/sap/ZGW_PO_SRV/" />
</app:service>
```

- Message Status 2* is Success message, 5* is Error message