How To...Subscription and Notification with SAP NetWeaver Gateway – OData Channel

Applicable Releases:
SAP NetWeaver Gateway 2.0 SP3

Version 1.0
March 2012
SAP “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java®. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java® Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td>&lt;<strong>Example text</strong>&gt;</td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Caution Icon" /></td>
<td>Caution</td>
</tr>
<tr>
<td><img src="image" alt="Note or Important Icon" /></td>
<td>Note or Important</td>
</tr>
<tr>
<td><img src="image" alt="Example Icon" /></td>
<td>Example</td>
</tr>
<tr>
<td><img src="image" alt="Recommendation or Tip Icon" /></td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
Table of Contents

1. **Business Scenario** ................................................................................................................................. 1
   1.1 Non-Mobile Scenarios ......................................................................................................................... 1
   1.2 Mobile Scenarios ................................................................................................................................. 1

2. **Background Information** ......................................................................................................................... 2

3. **System Prerequisites** .................................................................................................................................. 3

4. **Components** ................................................................................................................................................ 4
   4.1 SAP Business Suite ............................................................................................................................... 4
   4.2 SAP NetWeaver Gateway ...................................................................................................................... 4
       4.2.1 Generation Tools ............................................................................................................................ 4
       4.2.2 Custom Development (OData channel) .......................................................................................... 5
       4.2.3 SAP Business Workflow .............................................................................................................. 5
       4.2.4 SAP Mobile Applications ............................................................................................................. 6

5. **Step-by-Step Procedure** ............................................................................................................................ 7
   5.1 Create a Gateway Object to Read or Query ......................................................................................... 8
       5.1.1 Create a Model Provider Class ...................................................................................................... 9
       5.1.2 Create a Data Provider Class (Runtime) ..................................................................................... 10
       5.1.3 Configure Service in Backend ................................................................................................... 11
       5.1.4 Register Service in Gateway ...................................................................................................... 12

   5.2 Configure the Backend .......................................................................................................................... 13
       5.2.1 Communications from the Backend to Gateway .......................................................................... 14
       5.2.2 Update Metadata Class ............................................................................................................ 20
       5.2.3 Update Data Provider (Runtime) Class ..................................................................................... 23
       5.2.4 Check if service is subscription & notification enabled ............................................................. 24
       5.2.5 Implement Code for triggering Notification ............................................................................. 25

   5.3 Configure Gateway ................................................................................................................................. 28
       5.3.1 Enable communication between Gateway and Backend ............................................................. 28
       5.3.2 Maintain Destinations for Client Notifications .......................................................................... 30

   5.4 Executing the scenario ............................................................................................................................ 32
       5.4.1 Create subscription ...................................................................................................................... 32
       5.4.2 Trigger Event .............................................................................................................................. 37
       5.4.3 Receive notification .................................................................................................................... 38
       4.4.4 Optional – List Subscription ....................................................................................................... 40
       5.4.4 Optional – Subscription Detail ..................................................................................................... 41
       5.4.5 Optional – Delete Subscription .................................................................................................... 42

6. **Appendix** .................................................................................................................................................. 43
   6.1 Tracing ..................................................................................................................................................... 43

7. **Appendix – Subscription Collection** ...................................................................................................... 46
1. Business Scenario

SAP NetWeaver Gateway offers the ability to send notifications to a device and user who have subscribed to a suitable Gateway service. This capability does not need to be mobility centric however for the purpose of this document we shall assume it is. The illustration below highlights the basic premise behind notifications with SAP NetWeaver Gateway. This capability is different to SUP’s Data Change Notification function and is not covered in this document. For this document we will discuss Gateway eventing in isolation however the positioning of SUP will be explained in detail as there are clear benefits when consider this scenario in the context of mobility.

1.1 Non-Mobile Scenarios

When explaining subscription and notification using SAP NetWeaver Gateway is it important to note that your clients may not be mobile devices but applications or servers capable of consuming Odata services. With non-mobile scenarios, SUP is not a required component however an intermediate component or application is required to accept and manage subscriptions and notifications from consumers. The diagram below illustrates this valid scenario giving you control over your consumers and beyond mobile devices. The configuration How-To guides mentioned in the „Background Information“ chapter cover this scenario agnostic of mobility.

1.2 Mobile Scenarios

The inclusion of SUP/ODP in this architecture is specific to mobile device scenarios and offers significant features to support the subscription and notification process. SAP NetWeaver Gateway subscription and notification involves the following components when considering mobile scenarios. They independently need to be configured appropriately for your chosen scenario.
2. Background Information

This document gives you step-by-step configuration instructions for configuring SAP NetWeaver Gateway and the Backend for use with subscription and notification of services created with the OData Channel (programmatic).

The following additional How-to Guides and code samples are available for scenarios involving subscription & notification with SAP NetWeaver Gateway:

Overview
- How-To... Subscription & Notification with SAP NetWeaver Gateway - Overview

Gateway Configuration
- How-To... Subscription & Notification with SAP NetWeaver Gateway – Generation Tools
- How-To... Subscription & Notification with SAP NetWeaver Gateway – Custom Development (OData Channel)
- How-To... Subscription & Notification with SAP NetWeaver Gateway – Workflow

Client Development & SUP Configuration
- How-To... Subscription & Notification with SAP NetWeaver Gateway – iOS + Code Sample
- How-To... Subscription & Notification with SAP NetWeaver Gateway – Android + Code Sample
- How-To... Subscription & Notification with SAP NetWeaver Gateway – Blackberry + Code Sample

The following SAP Help links form the basis for this document and can be referenced in each section or centrally as per the list below:
- SAP NetWeaver Gateway 2.0 SP3 – Development Guide
- Odata Channel Introduction
- Subscription and Notification Flow - OData
  [http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm](http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm)
- Subscription and Notification Flow for Push Oriented Scenarios
  [http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm](http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm)
- Subscription and Notification Flow for Pull Oriented Scenarios
  [http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm](http://help.sap.com/saphelp_gateway20sp03/helpdata/en/92/10c17a436b44b7a4f74e30aac2cfef9/frameset.htm)
3. System Prerequisites

The following are required to be installed or ready before using this How-To Guide:

- You have access to a NetWeaver 7.02 SP7 or higher system with the SAP NetWeaver Gateway ABAP add-ons have been installed.
- The SFLIGHT Application needs to be configured and data tables populated by running the standard report SAPBＣ_DATA_GENERATOR.
- A SAP ERP system (referred as backend in this document). Corresponding Gateway 2.0 packages installed.
- Trust relationship has already been setup between Gateway and the backend. For detail on how to setup trust relationship, please refer to the online manual below:
  http://help.sap.com/saphelp_nw70ehp2/helpdata/en/f8/18da3a82f9cc38e10000000a114084/frameset.htm
- Firefox with REST Client installed for testing the RESTful services.
- A HTTP trace tool to capture the notification posted from the Gateway Server.

For more information on the steps mentioned in this document and other development activities please reference the official Gateway Development Guide.

For more information on the Odata Channel generally please refer to SDN for appropriate How-To Guides and supporting information. The following documentation links will be of help and should be referenced frequently. We assume you have basic Odata channel knowledge and can run the sample SFLIGHT services.

- SAP Official Documentation: http://help.sap.com/content/documentation/netweaver/docu_nw_gateway_design.htm
- SAP NetWeaver Gateway on SDN: http://www.sdn.sap.com/irj/sdngateway
- SAP NetWeaver Gateway How-To Guides: http://www.sdn.sap.com/irj/sdngateway?rid=/webcontent/uuid/coa638do-8478-2e10-8eb4-f157b64fb221
4. Components

4.1 SAP Business Suite

The SAP Business Suite represents the business logic or application layer responsible for holding the actual business data. SAP ERP, CRM & SRM are examples of SAP Business Suite applications and represent the core location for the application logic. In the context of Gateway and RESTful services objects such as Purchase Order, Employee and Material can be exposed to allow various clients to interact using Create, Read, Query, Update and Delete methods. Developers and architects need to choose between generating Gateway services off existing entities such as BOR/RFC & Screen Transactions or implement a programmatic service leveraging the OData Channel framework. See Design Decisions and Variations for more information.

4.2 SAP NetWeaver Gateway

SAP NetWeaver Gateway is a REST enabler that is available as a SAP NetWeaver Application Server ABAP (AS ABAP) add-on, which you can install on top of your existing SAP Business Suite or application platform. SAP NetWeaver Gateway offers development and generation tools to create OData services to a variety of client development tools. This is the vehicle to connect SAP Business Suite data and functionality to the target clients, platforms and programming framework. In this document we shall describe the subscription and notification scenario with a standalone Gateway instance integrating with a separate back-end Business Suite application server. Before we discuss the process of subscription and notification we need to introduce what type of Gateway services support subscription and notification.

The starting point for using these scenarios is the gateway service itself. Are you creating one from scratch or has the service already been delivered? Please follow the separate How-To guide showing the step-by-step procedure for the configuration. Once this is done a service is a service and follows the same method of creating the subscription. Some comments on each approach:

![Flowchart](chart.png)

- **Generation Tools**: Subscription and notification scenarios with SAP NetWeaver Gateway can be achieved using the generation tool in conjunction with Business Object Repository objects (BOR) only. The generation tool allows developers to build Gateway services against existing objects such as RFC, Business Object Repository methods and SAP Screen transactions. In this scenario the Gateway Service is built on the hub centrally providing clear abstraction to any Backend.

- **Customer Development (OData Channel)**: It is recommended to follow the customer development approach to gain full flexibility generally but also in the context of subscription and notification. In future capability we will see the introduction of a rich
development tool known as the Service Builder. This will provide a toolset with integration capabilities for code generation against existing assets locally and in other systems. i.e. RFC, BOR, BOL.

4.2.2 Custom Development (OData channel)

SAP NetWeaver Gateway services can be created with customer development. This is known as the OData Channel, and is a programmatic approach. This is the recommended approach to create Gateway services as it provides complete flexibility. It involves the creation of a model class for service definition and a runtime class with some additional registration. Both classes inherit specific super classes and required the developer to implement (redefine) the runtime methods associated with Create, Read, Update, Query and Delete. It really is a case of 1-2 lines of code to make an existing entity subscribable.

Subscription and notification is fully supported here and notifications are triggered with a simple code api located in the same system as where you develop your OData Service. Remember, you can deploy the required IW_BEP software component in the Gateway Hub system or a Backend and so you need to use the trigger API located locally. This is highlighted in the diagram below and is an example of such a deployment option. Business Object Repository eventing is also possible in this scenario by leveraging the event linkage table as normal however pointing to a custom class which can then call the standard OData Channel notification API.

The diagram below shows one deployment option when using custom development (OData Channel) with the required IW_BEP software component located in the Backend. Depending where you install the mandatory software component IW_BEP will determine how you can use subscription and notification along with where you code must be executed. More information on these differences can be found here.

4.2.3 SAP Business Workflow

SAP Business Workflow scenarios are also supported in the context of Subscription and Notification with SAP NetWeaver Gateway. Deliver services provide the capability to send notifications to task requiring a user to make decisions on and then send a response to the SAP and the appropriate backend workflow process. For more information on this capability please reference the How-To guide “Subscription and Notification with SAP NetWeaver Gateway – Workflow”.

April 2012
4.2.4 SAP Mobile Applications

SAP Mobile Applications, particularly the productivity applications, do support subscription and notification. Whilst this use case is not explicitly shipped with Gateway as an installation is it important to identify that subscription and notification is being used by some standard applications. For more information on these applications please refer to the associated help bound to the application it mobilizes. E.g. ERP Human Capital Management – Leave Requests etc. Click here for more information on these applications.
5. Step-by-Step Procedure

The scenario detailed in this document has four major configuration steps. The first is a pre-requisite to ensure you have the sample SFLIGHT gateway service running in your system. There are a couple of variations of this service depending on the SP level of your Gateway instance. Following this we move into the backend to configure the business object and event creation. This document focuses on the configuration in Gateway and the Backend only. For configuring SUP and mobile clients for use with this scenario please execute the steps in this guide and then refer to the How-To guide specific to your required client. E.g. Blackberry, iOS and Android. In each of these guides explicit instructions are given showing SUP configuration and the integration points between all components.

In our example we will use the RMTSAMPLEFLIGHT Service, which interacts with the SFLIGHT application found in every SAP instance. You should be familiar with this application and be able to run the sample Gateway service. After this step we configure the gateway application to accept subscriptions and persist for later notification back to a given client. Finally the last step is test the scenario using a REST client to create a subscription, a sample notification trigger program sending an event into Gateway and onto a HTTP trace tool to capture the notification.

<table>
<thead>
<tr>
<th>Step</th>
<th>Client</th>
<th>Gateway</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Gateway Object to read a Backend Business Object</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Configure the Backend</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>• Communications from the Backend to Gateway</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Update Metadata Class</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Update Data Provider (Runtime) Class</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Implement Code for triggering Notification</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configure Gateway</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>• Enable communication between Gateway and Backend</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Maintain Destinations for Client Notifications</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execute the Scenario</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>• Create Subscription</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Trigger Event in Backend</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Receive Notification</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Optional - List Subscription</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.1 Create a Gateway Object to Read or Query

It is important to reiterate that push notifications and subscription is dependent on an existing gateway object, which implements either a “read” or “getlist” method of an object. This document does not detail these steps however the following table summarizes the approach.

<table>
<thead>
<tr>
<th>Step</th>
<th>Client</th>
<th>Gateway</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a Gateway Object to READ or QUERY using the Odata Channel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Configure Backend Connectivity (ERP/Gateway)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Create a Model Provider Class (ERP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Create a Data Provider / Runtime Class (ERP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Configure Service in Backend (ERP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Register Service in Gateway (Gateway Registration)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the purpose of this document we will use the SFLIGHT application, which is shipped to all gateway installations. They will be referenced in subsequent sections and hence are worth noting.

<table>
<thead>
<tr>
<th>Object</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway Service</td>
<td>RMTSAMPLEFLIGHT</td>
</tr>
<tr>
<td>Backend System</td>
<td>ECC</td>
</tr>
<tr>
<td>Based on Generation Type?</td>
<td>Odata*</td>
</tr>
<tr>
<td>Metadata Class</td>
<td>/IWBEP/CL_MGW_MED_SFLIGHT</td>
</tr>
<tr>
<td>Data Provider (Runtime) Class</td>
<td>/IWBEP/CL_MGW_RT_SFLIGHT</td>
</tr>
</tbody>
</table>

Sample URL for Application
http://server:port/sap/opu/odata/sap/RMTSAMPLEFLIGHT?sap-client=100&$format=xml

The following screen shots summarize the objects listed above give a guide to the expected configuration you should have in your system. Again this guide assumes you have basic knowledge in creating Odata Channel Gateway Services and expect RMTSAMPLEFLIGHT to be running for comparison. To enable your bespoke Odata Channel Gateway Service specific instructions will be given highlighting the additions to be made.
5.1.1 Create a Model Provider Class

This class defines the schema or metadata for a given Odata Service. The key method here is DEFINE and the resulting source code can be seen in the next graphic. Remember for SFLIGHT these classes are shipped with package IW_BEP and hence are already available. For your own development you will need to implement this type of class yourself. **When you create your own Odata Service you will inherit the super class /IWBEP/CL_MGW_ABS_MODEL however as you will see later we need to change this to be /IWBEP/CL_MGW_PUSH_ABS_MODEL.**

Example code from SFLIGHT method DEFINE. We will highlight the important lines of code later.

```
METHOD define.

DATA:
  ls_assocation TYPE REF TO /ibew/if OverwatchAssociation,
  ls_data_object TYPE REF TO /ibew/if OverwatchDataObject,
  ls_flight_object TYPE REF TO /ibew/if OverwatchFlightObject,
  ls_pojko TYPE REF TO /ibew/if OverwatchPojko,
  ls_storage_object TYPE REF TO /ibew/if OverwatchStorageObject,
  ls_field_name TYPE REF TO /ibew/if OverwatchFieldName,
  ls_key_property TYPE REF TO /ibew/if OverwatchKeyProperty;

  * Set the header element
  * It is a key element.
  * The method also uses different names for the fields in the data structure (lr_flight_key)
  * lr_flight_key is used to allow faster access to the fields in the object.
  * lr_flight_fieldname parameter: lr_flight_fieldname must be used according to the concrete
  * lr_flight_fieldname = lr_flight_object-recoctance_property('EINRICHT')
  * lr_flight_fieldname = lr_flight_object-recoctance_property('EINRICHT')
  * lr_flight_fieldname = lr_flight_object-recoctance_property('EINRICHT')
  * lr_flight_fieldname = lr_flight_object-recoctance_property('EINRICHT')

  lr_flight_key TYPE REF TO /ibew/if Overwatch_flight_key;
  lr_field_name TYPE REF TO /ibew/if Overwatch_fieldname;
  lr_key_property TYPE REF TO /ibew/if Overwatch_key_property;
  lr_property TYPE REF TO /ibew/if Overwatch_property;

  * First creating the flight data object
  * lr_flight = CREATE AS lr_flight_data_object('FLIGHT');

  * Using the addDelete interface for inserting the properties in a bulk manner instead of
  * lr_property = lr_flight_object-recoctance_property('LASTEDIT');

  * Set it as a key element.
  * lr_property = lr_flight_key_lr_key;

  * lr_flight_assocation lr_flight_key lr_key_property lr_field_name lr_property lr_field_name lr_key_property lr_property lr_field_name lr_key_property lr_property
```

April 2012
5.1.2 Create a Data Provider Class (Runtime)

This class is responsible for implementing the business logic for a given Gateway service. The developer is responsible for re-defining the methods required (GET_ENTITY) etc., relative to the CRUD methods. Remember for SFLIGHT these classes are shipped with package IW_BEP and hence are already available. For your own development you will need to implement this type of class yourself. This document assumes you have basic knowledge in creating Odata Channel Gateway services and are familiar with these steps. When you create your own Odata Service you will inherit the super class /IWBEP/CL_MGW_ABS_DATA, however as you will see later we need to change this to be /IWBEP/CL_MGW_PUSH_ABS_DATA.

The sample code below is from the RMTSAMPLEFLIGHT Service and shows the code executed when a READ is expected. In real life you would implement the methods required for your own service. Remember this is where the business logic lives to get the data our of the application layer. i.e MATNR VBAK etc.
5.1.3 Configure Service in Backend

After creating the two classes a developer needs to create some configuration entries, which bind the two together when used in the context of a Gateway Service. The graphic below depicts this configuration. For SFLIGHT this configuration should already be available or can be easily created in your own namespace. This document assumes you have basic understanding of how to create an Odata channel Gateway service. You can find this configuration link in SPRO/IMG via SAP NetWeaver → SAP NetWeaver Gateway Business Suite Enablement → Backend Odata Channel → Maintain Models.

![Configuration Graphic]

You can find this configuration link in SPRO/IMG via SAP NetWeaver → SAP NetWeaver Gateway Business Suite Enablement → Backend Odata Channel → Maintain Service.
5.1.4 Register Service in Gateway

Once the classes and configuration have been created in the backend the developer needs to register these in Gateway so a service can be actually called and registered in SICF. The following screen shot depicts this process. Again this document assumes you have SFLIGHT (RMTSAMPLEFLIGHT) Configured and that you are familiar with Odata Channel Gateway Services. For more information on Odata Channel Services please refer to the appropriate How To guide. You can find this configuration link in SPRO/IMG via SAP NetWeaver → Gateway → Odata Channel → Administration → General Settings → Maintain Service.
5.2 Configure the Backend

For Odata Channel scenarios we must enable asynchronous communications between the Backend and Gateway. This will enable notifications to be sent using BGRFC and a configured SM59 RFC Destination. This section also requires us to configure the Supervisor destination and relevant configuration specific to BGRFC capabilities.

<table>
<thead>
<tr>
<th>Configure the Backend</th>
<th>Client</th>
<th>Gateway</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Communications from the Backend to Gateway</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Update Metadata Class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Update Data Provider (Runtime) Class</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Check if service is subscription &amp; notification enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Implement Code for triggering Notification</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
</tbody>
</table>

For a summary of transactions used in this section please see the Appendix: Transaction Map Backend for more information.

Software Requirements
The following are the minimum software requirements for using Backend Event Publisher in your SAP system:

- SAP NetWeaver Application Server ABAP 7.0, or higher
- SAP ECC 6.0, ECC-DIMP, 600, 0015 (SAPK-60015INECCDIMP), DIMP
- Make sure that you have administrator privileges in the SAP system.

Enabling Backend Event Publisher
You install the following installation packages to enable Backend Event Publisher:

- In your SAP system, install the component, /IW_BEP.
5.2.1 Communications from the Backend to Gateway

Before configuring the actual scenario we need to enable communications between Gateway and the SAP Backend (ECC). This is done using RFC Destinations and the bgRFC protocol. There are two destinations to set up in this section. They are both use background RFC as the protocol and represent one for the bgRFC Supervisor for local communication and another for the actual call to the Gateway Server. The following steps guide you through the configuration of these two destinations.

1. Log on to the Backend system (ECC). Go to transaction SM59.
2. Define Supervisor Destination BGRFC_SUPERVISOR with connection Type “3”.
3. Enter user, password for the logon info. Be reminded that the user you enter needs to at least have authorizations in the SAP_BC_BGRFC_SUPERVISOR role or is assigned with the role.
4. Go to transaction SBGRFCCONF. On the Scheduler App Server tab, click on Create.

5. Enter a server name with F4 help and save.

6. On Supervisor Destination tab, click on Change Button and enter the RFC Destination created in step Then press the save button.
In the next steps, you will need to configure the Backend Event Publisher (BEP) in the backend system and define the connection between the Backend and Gateway through bgRFC. At runtime the Backend Event Publisher will push events over this RFC Destination into the Gateway server. Note if you have configured other Gateway scenarios such as Workflow Notifications this RFC Destination may already be configured. Please consult your Administrator for more information.

1. Still in the backend system, go to transaction SPRO. Open the IMG structure SAP NetWeaver → SAP NetWeaver Gateway Business Suite Enablement → Backend Odata Channel → Connection Settings to SAP NetWeaver Gateway. Then click on the Activity icon next to Create RFC Destination for Outbound Queues.

2. In the window of Configuration of RFC Connections, click on the Create button.
3. This will bring us to the RFC Destination transaction (i.e. SM59). We now need to create a bgRFC destination. Enter the following to the corresponding fields and hit ENTER.
   - RFC Destination: GWDEMO_BGRFC_DEST <or any name you want>
   - Connection Type: 3
   - Description 1: bgRFC for Backend Event Publisher <or your own description>

4. Under the tab Technical Settings, enter the host info of the system where Gateway is located.
   - Target Host: <hostname of the Gateway Server>
   - System Number: <system # of the Gateway Server>

5. Click on the tab Logon & Security. Enter the Logon info for logging on to the Gateway Server. Click on Save to save the changes.

6. Click on the tab Special Options. Under the area of Select qRFC Version, select 1 bgRFC for qRFC Version.
7. Click Save. A dialog of “Warning about selection of qRFC version” will be displayed. Click on Yes to proceed.

8. On the bottom of the screen, you should see a message “Destination GWDEMO_BGRFC_DEST saved”. Click on the button Connection Test to test the destination. A successful test should look like below.

9. Go to transaction SPRO and open the IMG structure SAP NetWeaver → SAP NetWeaver Gateway Business Suite Enablement → Backend Odata Channel → Connection Settings to SAP NetWeaver Gateway. Click on the Activity icon next to SAP NetWeaver Gateway Settings.
10. In the window of Change View “Gateway Settings”: Overview, click on New Entries. Then enter the followings in the corresponding fields.
   - Destination System:  W64 < i.e. Gateway Server SID >
   - Client:  100 < i.e. Gateway Server Client >
   - System Alias:  W64_GWDEMO < an unique alias name >
   - RFC Destination:  GWDEMO_BGRFC_DEST < the bgRFC destination created in step 11 of this section >

11. Click on Save. Create your Customizing Request when prompted.
5.2.2 Update Metadata Class

Compared to the generation tool scenario the custom development option (Odata Channel) requires little change to your existing service. The following steps are done in the class editor (SE80 etc) and we simply change the inherited class and makes some additions in DEFINE method for your given entity. See below.

Please refer to the Class [/IWBEP/CL_MGW_MED_SFLIGHT] as an example implementation of a Subscription/Notification metadata definition.

1. Create a class for metadata definition. This has already been done as a pre-requisite for this document. We shall demonstrate the RMTSAMPLEFLIGHT classes as an example.

2. Inherit from the SAP NetWeaver Gateway object/meta model super class [/IWBEP/CL_MGW_PUSH_ABS_MODEL]. In the class properties simply click change and inherit this standard class. You can see an example below. For normal Odata model classes this would typically be [/IWBEP/CL_MGW_ABS_MODEL].
3. Redefine method DEFINE() in your class to create your own implementation. If you have created the class this would already be done as per basic Odata Channel methodology.

4. Add SUPER->DEFINE( ) as first statement to your implementation to define the subscription data for your application. This is really important. Now that we have inherited a different class we have to call the super. See the screen shot below for an example.

```java
METHOD define.
DATA:
  lo_annotation TYPE REF TO /iwbp/if_mgw_odata_annotation,
  lo_data_object TYPE REF TO /iwbp/if_mgw_odata_entity_up,
  lo_flight_object TYPE REF TO /iwbp/if_mgw_odata_entity_up,
  lo_entity_set TYPE REF TO /iwbp/if_mgw_odata_entity_set,
  lo_complex_type TYPE REF TO /iwbp/if_mgw_odata复杂类型,
  lo_association TYPE REF TO /iwbp/if_mgw_odata关联,
  lo_nav_property TYPE REF TO /iwbp/if_mgw_odata nav prop,
  lo_property TYPE REF TO /iwbp/if_mgw_odata property,
  lo_action TYPE REF TO /iwbp/if_mgw_odata action,
  lo_parameter TYPE REF TO /iwbp/if_mgw_odata parameter,
  lo_ref_constraint TYPE REF TO /iwbp/if_mgw_odata ref constr.

*call super method to include subscriptions/notifications
super->define().
```

5. Implement metadata definition as explained in Define your Object Model in the Backend System. This step just expects you to implement your schema definition as per normal. i.e. SFLIGHT has FlightCollection, BookingCollection etc.
6. Call method SET_SUBSCRIBABLE() for each entity type in your implementation for which you want subscription and notification to be enabled. This is really important as it determines exactly what in your service is subscribable. See the graphic below as an example.

```
lo_flight_object = model->create_entity_type('Flight').

* using the convenience interface for creating the properties in a bulk manner instead of calling create
  lo_property = lo_flight_object->create_property('c吊顶').
* set it as a key element
  lo_property->set_is_key().
* one could also use different names for the fields in the ABAP structure (iv_abap_fieldname) and extend
  by default external names are converted to uppercase so that the following would have the same result.
* iv_abap_fieldname parameter. iv_abap_fieldname must be used according to the constraints of a field
  lo_property = lo_flight_object->create_property( iv_property_name = 'CONNTID',
  iv_abap_fieldname = 'CONNTID').

lo_property->set_is_key().
lo_property = lo_flight_object->create_property('f吊顶').
lo_property->set_is_key().

lo_property = lo_flight_object->create_property('PRICE').
lo_property->set_unit('CURRENCY').
lo_property->set_sortable().

lo_property = lo_flight_object->create_property('CURRENCY').
lo_property->set_semantic(/nkiwsp/cl_mgw_abs_model->gcs_sap_semantic-currency_code ).
lo_property->set_sortable( abap_false ).

lo_property = lo_flight_object->create_property('PLANETYPE').
* Set text type to class text element
  lo_property->set_LABEL_FROM_TEXT_ELEMENT( iv_text_element_symbol = '000',
    lo_object_ref = me ).

lo_flight_object->create_property('SEATSIGNX').
lo_flight_object->create_property('SEATSIGNY').
lo_flight_object->create_property('PAYMENTSUN').
lo_flight_object->create_property('SEATSIGNX_S').
lo_flight_object->create_property('SEATSIGNY_S').
lo_flight_object->create_property('SEATSIGNX_F').
lo_flight_object->create_property('SEATSIGNY_F').

* a structure is bound to the data object, so all properties which are already defined or those
  which were newly created were matched to the meta data of the ddc via the abap names. Texts, type
  lo_flight_object->bind_structure( '/NWBE/CL_MGW_RT_SPLIGHT->TY_S_SPLIGHT' ).

* for push scenarios it is required to set an object as subscribable for getting notified on changes
  lo_flight_object->set_subscribable().
lo_flight_object->set_pachable().
lo_flight_object->set_addressable().
```
5.2.3 Update Data Provider (Runtime) Class

As per the metadata or model class we need to make a relative small change to the super class it inherits and also re-define a security method to perform your custom authority checks. The following steps are done in the class editor (SE80 or similar). Almost no change here!

Please refer to the Class /IWBEP/CL_MGW_RT_SFLIGHT as an example implementation of a Subscription/Notification metadata definition.

1. Create a class for data provisioning. This has already been done as a pre-requisite for this document. We shall demonstrate the RMTSAMPLEFLIGHT classes as an example.

2. Inherit from the SAP NetWeaver Gateway data provider super class /IWBEP/CL_MGW_PUSH_ABS_DATA. In the class properties simply click change and inherit this standard class. You can see an example below. For normal Odata model classes this would typically be /IWBEP/CL_MGW_ABS_DATA.

3. Redefine abstract method CHECK_SUBSCRIPTION_AUTHORITY in your class to implement your checks for subscription authorization. This method is an empty code space where the developers can implement their own checks to see if a user should be able to subscribe. Remember when using the Odata Channel approach you are in control of everything.

4. Redefine methods of interface /IWBEP/IF_MGW_APPL_SRV_RUNTIME in your class to process the operations of your application, for example creation or deletion of an entity. Implement the data provider as explained in Define your Runtime Class in the Backend System. This step merely states you should continue to implement the appropriate CRUD methods as per normal. You may already have done this.
5.2.4 Check if service is subscription & notification enabled

Once you have a created or enabled you service to be subscription and notification enabled the metadata document and service document should reflect the new capability. Below is an example of the service document from the famous SFLIGHT application, which is shipped with every Gateway instance. You can see the service document indicates it is subscription and notification enabled and that the FlightCollection can be subscribed against.

Below is an example of the metadata document from the famous SFLIGHT application, which is shipped with every Gateway instance. You can see the metadata document indicates it is subscription and notification enabled and actually shows the schema of the each. That tells us what an inbound notification would look like and how the client API can easily covert to consumable data types. Conversely it also shows us what a subscription request would look like from a payload perspective. We will use this when we create the subscription highlighted later in this document.

It is important to note that the Subscription Collection acts like any other Collection. You can create, update, delete, read and query any subscriptions using the HTTP Verb combination with an appropriate URL and payload. For more information please see „Appendix – Subscription Collection“ for more information.
5.2.5 Implement Code for triggering Notification

Compared to the generation tools scenario the customer development option (OData Channel) is programmatic and requires the developer to explicitly trigger notifications. This approach gives flexibility and allows the developer to inject the notification trigger where they see fit. Such a place would be a user exit or BADI.

In the system the ABAP report /IWBEPR_MGW_PUSH_TEST is available to demonstrate the notification transmission.

1. Call static method /IWBEPR_MGW_SUB_REGISTRY=>GET_SUBSCRIPTION_REGISTRY( ) to get the instance of the subscription registry.
2. Call method GET_SUBSCRIPTIONS() of the subscription registry providing the model group name and the collection for the notifications of the current business application.

3. Call static method /IWBEPIWBEPCL_MGW_NOTIF_PUBLISHER=>GET_NOTIFICATION_PUBLISHER() to get the instance of the notification publisher.
4. Call method `CREATE_NOTIFICATION_ENDPOINTS()` of the publisher instance providing the data structure you want to send and the list of subscriptions returned by the registry.

5. Call method `SEND_NOTIFICATIONS()` of the publisher for each notification endpoint.
5.3 Configure Gateway

Now the Backend system has been configured we must turn our attention to the Gateway server and complete configuration. As we have already registered the ERP gateway service there is little to do. This configuration is limited to RFC Destinations and IMG Gateway settings. Note the basic communication between Gateway and ERP should already have been done given you have the RMTSAMPLEFLIGHT service operational. The table below highlights the approach.

<table>
<thead>
<tr>
<th>Configure Gateway</th>
<th>Client</th>
<th>Gateway</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Enable communication between gateway and backend</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>• Maintain destination for client notifications</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For a summary of transactions used in this section please see the Appendix: Transaction Map Gateway for more information.

5.3.1 Enable communication between Gateway and Backend

The first step is to configure the RFC communications between Gateway and the Backend (ECC). This is done through standard RFC Destination in SM59 followed by an Alias definition in the Gateway IMG configuration.
1. Logon to the Gateway Server. Then go to transaction SM59. You will need to create a RFC destination to the backend system. This may already be created so you should check existing Alias definitions and underlying RFC Destinations. Consult your Administrator for more information. Create a destination with the info below. Save and test the connection to make sure it works.
   - RFC destination: ERPCLNT400 < or any unique name you want >
   - Connection Type: 3
   - Target Host: < SAP Backend Hostname >
   - System Number: < SAP Backend Sys Number ##>

2. Go to transaction SPRO and open the IMG Structure SAP NetWeaver Gateway → Gateway → OData Channel → Configuration → Connection Settings → SAP NetWeaver Gateway to SAP System. Click on the Execute icon next to Manage SAP System Aliases.

3. Click on the button New Entries. Define a SAP system alias with info below:
How To...Subscription and Notification with SAP NetWeaver Gateway - Custom Development (OData Channel)

- SAP System Alias: ERP_400 < or any unique name you want >
- Description: < any description >
- RFC Destination: < choose the RFC destination defined in step 1 >

4. Click on Save.

5.3.2 Maintain Destinations for Client Notifications

Before testing the scenario we need to maintain the destination from which a business object event notification will be sent to. This is a security measure and allows central configuration of not only the destination but also the authentication requirements specific to a single destination.

1. Go to transaction SM59 or use the navigation path seen below. SAP NetWeaver → Gateway → Generic Channel → Configuration → Subscription Settings and then click Maintain RFC Destination to Connectivity Provider.

2. Create a new destination of type G “HTTP to External Server” and maintain the URL, Service and Log on information. Click Save. This could point to your tracing program or http listener. In real world environment this would point to the device infrastructure, which would then forward to any subscribed device etc.
This destination is important, as it will route notifications to a connectivity provider or listening application. In a mobile scenario we would explicitly configure the SM59 destination to route to SUP with a set Path Prefix and Service. The image below shows this configuration for mobile scenarios. Explanation of mobile scenario routing is detailed in document “How To ... enable subscription and notification with NetWeaver Gateway – Overview”.

![Diagram of RFC Destination](image-url)
5.4 Executing the scenario

After completing the configuration described in the previous section we are now ready to test the scenario. The table below highlights the approach.

Note – This document makes no assumptions about your client and hence to execute this scenario we shall leverage the Firefox REST client. In a productive environment this would be a mobile client or application, which programmatically calls these services.

<table>
<thead>
<tr>
<th>Execute the Scenario</th>
<th>Client</th>
<th>Gateway</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Subscription</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Trigger Event in Backend</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Receive Notification</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Optional - List Subscription</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Optional - Delete Subscription</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

5.4.1 Create subscription

To issue a subscription request we will use the REST client for Firefox as a test harness for this. Start Firefox, click on Tools → REST Client. In the drop-down box of Method, choose POST. Then enter a URL with the format below in the text field:

**Syntax:**

http://<Gateway hostname>:<Gateway host http port> /sap/opu/sdata/sap/<your service document>/SubscriptionCollection

**Example:**

http://10.1.1.50:8042/sap/opu/sdata/sap/RMTSAMPLEFLIGHT/SubscriptionCollection

Click on the Login button and enter your username and password to login to the Gateway Server. This will add an Authorization header to the request.

For SP2 Click on the Add Request Header button. Add a header:

- Name: x-requested-with
- Value: XMLHttpRequest

For SP3 there are additional requirements around HTTP headers and tokens. For more information please see this link:


In the Request Body area, enter an XML like below. If you need more information on these elements look at the metadata document or the following link.
How To...Subscription and Notification with SAP NetWeaver Gateway - Custom Development (OData Channel)


The following table describes the most important XML elements and their meaning in relation to our scenario:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>This can be any text</td>
</tr>
<tr>
<td>deliveryAddress</td>
<td>Specifies the location of subsequent notifications for Business Object events. We will put a trace on this location to demonstrate what is posted in when a Backend Business Object event is created. At runtime this URL and port is compared to the HTTP destinations defined in SM59. If there are two or more SM59 destination it will retrieve the first entry only. For mobile scenarios with SUP this URL can be retrieved dynamically from SUP application connection properties set by an administrator.</td>
</tr>
<tr>
<td>Collection</td>
<td>The Gateway Business Object Collection created in the pre-requisite scenario. If you are unsure of this name you can always search for it in the metadata document available by performing a HTTP Get on http://&lt;Gateway hostname&gt;&lt;Gateway host http port&gt;/sap/opu/sdata/sap/&lt;your service document&gt;/metadatathefilter notation and supports only AND operators. E.g If the filter is left empty we will subscribe to all material creation events. We can put something like the following to subscribe only to a specific material. The following is an example of Sales Order. E.G. &lt;d:filter&gt;value eq '0000004003' and scheme_id eq 'SALES_ORDER' and scheme_agency_id eq 'Q5K_004'&lt;/d:filter&gt;</td>
</tr>
<tr>
<td>filter</td>
<td>A filter enables the user to either subscribe to all business object instances or only a specific entity instance. This complies to the normal Odata $filter notation and supports only AND operators. E.g If the filter is left empty we will subscribe to all material creation events. We can put something like the following to subscribe only to a specific material. The following is an example of Sales Order. E.G. &lt;d:filter&gt;value eq '0000004003' and scheme_id eq 'SALES_ORDER' and scheme_agency_id eq 'Q5K_004'&lt;/d:filter&gt;</td>
</tr>
</tbody>
</table>
select

Either * or blank. It is possible to retrieve the object content instead of just the event information. This is the responsibility of the application in the backend to populate the entire payload or business object contents. Please consult the backend application documentation to understand if this is supported.

persistNotifications

This element determines if notifications are sent (pushed) to a client or whether they are stored for later retrieval via a pull. To retrieve notifications via pull you need to call a different service all together. Example:


Note – This is only supported when implementing subscription and notification when using the OData channel and NOT the generation tool option.

2. Then click on the Send button to post the subscription request. You should receive a HTTP 201 Created response code, which can be seen in the bottom window detail section. If you do not receive such a response code you may have a subscription already created or there is an issue with the configuration done in the previous sections. Please see the Appendix section “Monitoring and Troubleshooting” for more information.
Note:
If you get an error of “Subscription already exists” from an HTTP response and you want to remove the old subscription please see the List Subscription and Delete Subscription sections following the Receive Notification section.

3. To prove you have created a subscription in the Backend ECC Server go to transaction SE11 and view the contents of table /IWBEIP/D_MGW_SUB. You should see an entry representing your newly created subscription.
How To...Subscription and Notification with SAP NetWeaver Gateway - Custom Development (OData Channel)

Dictionary: Display Table

Data Browser: Table /IWBEP/D_MGW_SUB Select Entries

Table: /IWBEP/D_MGW_SUB
Displayed Fields: 6 of 10
Fixed Columns: 2
List Window 0250

Table /IWBEP/D_MGW_SUB Display

April 2012
5.4.2 Trigger Event

Now that we have created a subscription successfully and verified it has been propagated from the REST client to Gateway and into the relevant Backend, we can trigger the notification using the report. Remember this report or code is to be integrated into your standard processing within the SAP application server via user exit or BADI etc.

1. Log into the Backend and execute your customer report or for the SFLIGHT application you can specifically use /IWBEP/R_MGW_PUSH_TEST

![ABAP Editor: Initial Screen](image1)

![Report /IWBEP/R_MGW_SFLIGHT_RAISE_EVT](image2)
5.4.3 Receive notification

A notification will be sent from Gateway to the URL specified in the original subscription create HTTP POST. For testing purposes you could run a trace on the specified location, which would be a web server etc. In our scenario I have used a simple HTTP trace program implemented in java. See below for setting up this program. In a real world scenario this would be a mobile platform such as SUP or an application. For more information on how to set up SUP in such scenarios please see the device specific guide such as “How-To.. Subscription and Notification with SAP NetWeaver Gateway – Android”. We have guides for iOS and Blackberry.

1. Download the a HTTP Trace program such as http://httptrace.sourceforge.net/httptrace.zip

2. Unzip the files to a location which can has access to the java executable. SAP takes no responsibility for this application and recommends consulting your system administrator for compliance with organizational policies. You could simply create a subscription with the deliver address element pointing to a web server and set up tracing or logging on your own web server.

3. Run the HttpTrace java program and configure the trace to listen on a free port and forward to an existing web server. Note the free port must correspond to the DeliveryAddress element in your subscription creation HTTP Post done earlier.

4. Once a notification has been sent at runtime you will see it hit the trace program and display as per the following.
5. To demonstrate the content of the notification the xml has been saved to a separate file and can be seen below. The notification seen below shows material but is the same structure for all notifications. For SFLIGHT you would see a different title and collection Id.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:IWF="http://www.sap.com/Protocols/IWFData">
    <atom:content type="application/xml">
        <m:properties>
            <m:value d:id="0019BCB815E1EE060408888DF9201C">
                <d:scheme_id="IWF_NOTIFICATION">
                    <d:scheme_agent_id="LOCAL">
                        <d:collection:"2ASwMaterialCollection"/>
                    </d:scheme_agent_id>
                </d:scheme_id>
            </m:value>
        </m:properties>
    </atom:content>
</atom:entry>
```
4.4.4 Optional – List Subscription

To get a list of subscriptions issue a GET request to the following URL:

**Syntax:**

http://<Gateway hostname>:<Gateway host http port>/sap/opu/sdata/sap/<service document>/SubscriptionCollection

**Example:**

http://10.1.1.50:8042/sap/opu/sdata/sap/RMTSAMPLEFLIGHT/SubscriptionCollection

You should receive an xml representation of your subscriptions similar to below.
5.4.4 Optional – Subscription Detail

To retrieve detail of an existing subscription please follow the below syntax in the REST client with the HTTP method of GET.

Syntax:

http://<Gateway hostname>:<Gateway host http port>/sap/opu/sdata/sap/<service document>/SubscriptionCollection(ID='[Subscription GUID ]')

Example:

http://usphlrig15.phl.sap.corp:8001/sap/opu/sdata/sap/RMTSAMPLEFLIGHT/SubscriptionCollection(ID='0019BBC8B15E1EE08FDFF5A31853201C')

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription ID</td>
<td>The actual GUID of your subscription. This is the one in the SE11 table mentioned before and can also be retrieved by using the collection method with GET shown in the previous section.</td>
</tr>
</tbody>
</table>
5.4.5 Optional – Delete Subscription

To delete an existing subscription please follow the below syntax in the REST client with the HTTP method of DELETE.

Syntax:

http://<Gateway hostname>:<Gateway host http port>/sap/opu/sdata/sap/<service document>/SubscriptionCollection(ID='[Subscription GUID ]')

Example:

http://usph1rig15.phl.sap.corp:8001/sap/opu/sdata/sap/RMTSAMPLEFLIGHT/SubscriptionCollection(ID='0019BBC8B15E1EE08FDFF5A31853201C')

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscription ID</td>
<td>The actual GUID of your subscription. This is the one in the SE11 table mentioned before and can also be retrieved by using the collection method with GET shown in the previous section.</td>
</tr>
</tbody>
</table>

1. The resulting HTTP response should be a 204 No Content as seen from the screenshot below. Note you can check the tables in both ECC and Gateway to test if the subscription has been deleted.
6. Appendix

6.1 Tracing

When testing your scenario is a possibility that a configuration step may not have been completed or the scenario just does not work. If this does occur we recommend increasing the trace configuration settings so you can see what is happening at runtime. For Gateway information please use the standard application log with transaction SLG1. The following procedure highlights how to increase tracing or logging for Gateway runtime diagnostics.

Go to transaction SPRO. In the toolbar click on SAP Reference IMG.

2. Expand the IMG structure to the find the Gateway Subscription setting via the following path Cross-Application Components → Project Gateway → Project Gateway Administration → Logging Settings. Click on the Set Log level for All Users.
3. For each entry in the table change the logging to the required level.

4. Test your scenario again and look at the application log with SLG1 for more information.
### Display logs

<table>
<thead>
<tr>
<th>Date/Time/User</th>
<th>Nu</th>
<th>External ID</th>
<th>Object ID</th>
<th>Sub-object ID</th>
<th>Tran</th>
<th>Prog</th>
</tr>
</thead>
<tbody>
<tr>
<td>28.02.2011 19:29:27 CHANCH</td>
<td>23</td>
<td>No request ID</td>
<td>Service Consum.</td>
<td>SCL Runtime P.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28.02.2011 19:29:27 CHANCH</td>
<td>23</td>
<td>No request ID</td>
<td>Service Consum.</td>
<td>SCL Runtime P.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Problem class Additional info</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- All messages of this request will be logged
- Started processing of event GW_BUS1001_CREATED for object IMC_0014_ANDY10 received from E05/400
- The process of adding an event to the Event Manager has started
- The Notification has event type IWFDICL_SUB_EVT_NOTIFICATION and is for object type IWF_NOTIFICATION
- The Notification has been added to the queue
- The process of adding an event to the Event Manager has been completed
- Deferred processing of events has been initiated
- Processing of the event has started
- The Notification has event type IWFDICL_SUB_EVT_NOTIFICATION and is for object type IWF_NOTIFICATION
- No request ID provided. Idempotent functionality not used.
- The process of calling the Event Handler IWFDICL_SUB_NTFT_EVT_HANDLER has started
- Consumer factory: init for gado:service_id IWF_NOTIFICATION
- Consumer constructor: Compo IW_PUSH, Root NOTIFICATION, gado IWF_NOTIFICATION, ConsID
- Consumer constructor was executed successfully
- Consumer factory: end for Object
- Create adapter process has started in the adapter factory class
- Consumer adapter class instance has been created (class name: IWFDICL_SUB_CNP_SDF_ADAPTER)
- Create adapter process has been completed in the adapter factory class
- The process of calling the Event Handler IWFDICL_SUB_NTFT_EVT_HANDLER has been completed
- Processing of the event has been completed
- Deferred processing of events has been completed
- Completed processing of event GW_BUS1001_CREATED for object IMC_0014_ANDY10 received from E05.
- The request has been successfully processed on the SCL server
7. Appendix – Subscription Collection

The subscription collection bound to you enabled Gateway service follows the same principal as any Gateway service. This means that a user or application can Query, Create, Read, Update and Delete subscriptions simply by using the combination of URL, HTTP verb and payload.

<table>
<thead>
<tr>
<th>Action</th>
<th>HTTP Verb</th>
<th>Request Payload</th>
<th>Sample URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>GET</td>
<td>Empty</td>
<td>.../SubscriptionCollection</td>
</tr>
<tr>
<td>Read</td>
<td>GET</td>
<td>Empty</td>
<td>.../SubscriptionCollection('ID=1323098ASDAS')</td>
</tr>
<tr>
<td>Create</td>
<td>POST</td>
<td>Subscription Entity</td>
<td>.../SubscriptionCollection('ID=1323098ASDAS')</td>
</tr>
<tr>
<td>Delete</td>
<td>DELETE</td>
<td>Empty</td>
<td>.../SubscriptionCollection('ID=1323098ASDAS')</td>
</tr>
<tr>
<td>Update</td>
<td>PUT</td>
<td>Subscription Entity</td>
<td>.../SubscriptionCollection('ID=1323098ASDAS')</td>
</tr>
</tbody>
</table>