



SAP Gateway Foundation Developer Guide

PDF download from SAP Help Portal:

http://help.sap.com/saphelp_nw74/helpdata/en/a6/422751c639276ee10000000a445394/content.htm

Created on August 02, 2016

The documentation may have changed since you downloaded the PDF. You can always find the latest information on SAP Help Portal.

Note

This PDF document contains the selected topic and its subtopics (max. 150) in the selected structure. Subtopics from other structures are not included. The selected structure has more than 150 subtopics. This download contains only the first 150 subtopics. You can manually download the missing subtopics.

© 2016 SAP SE or an SAP affiliate company. All rights reserved. No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE. The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary. These materials are provided by SAP SE 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. SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE in Germany and other countries. Please see www.sap.com/corporate-en/legal/copyright/index.epx#trademark for additional trademark information and notices.

Table of content

- 1 SAP Gateway Foundation Developer Guide
 - 1.1 OData Channel
 - 1.1.1 OData in SAP Gateway - Feature Overview
 - 1.1.1.1 OData Query Options
 - 1.1.2 General Considerations
 - 1.1.2.1 OData Best Practices
 - 1.1.2.2 Performance Best Practices
 - 1.1.2.3 Creating High-Quality OData Services
 - 1.1.2.4 Deployment Considerations for Development
 - 1.1.2.4.1 Development in SAP Gateway Hub System
 - 1.1.2.4.2 Development in SAP Business Suite Backend System
 - 1.1.3 Basic Features
 - 1.1.3.1 Catalog Service
 - 1.1.3.2 Media Links
 - 1.1.3.3 Service Life-Cycle
 - 1.1.3.3.1 Activate and Maintain Services
 - 1.1.3.3.2 Maintaining Models and Services
 - 1.1.3.3.3 Maintain Annotation Models
 - 1.1.3.3.4 Metadata Cache
 - 1.1.4 Advanced Features
 - 1.1.4.1 Service Tagging
 - 1.1.4.2 ETag Handling
 - 1.1.4.2.1 ETag Handling in Batch Processing
 - 1.1.4.3 \$batch Processing
 - 1.1.4.4 Patch Support
 - 1.1.4.5 Delta Query Support
 - 1.1.4.6 Conversion Handling in OData Channel
 - 1.1.4.7 Expand in Framework and Data Provider
 - 1.1.4.8 Deep Insert
 - 1.1.4.9 Error Response Control for Backend Data Provider
 - 1.1.4.10 \$filter System Query Option APIs
 - 1.1.4.11 Map Message Container to Message Protocol Format
 - 1.1.4.12 Annotations
 - 1.1.4.13 Extended Support of Long Texts in the Metadata
 - 1.1.4.14 Soft State Support for OData Services
 - 1.1.4.14.1 Soft State Based Query Result Cache
 - 1.1.4.15 Search and Open Search Capabilities
 - 1.1.4.16 Simple Field Extensibility
 - 1.1.4.17 Conversion Support for Complex Filters
 - 1.1.4.18 Excel Support
 - 1.1.4.19 User Self Service
 - 1.1.4.19.1 Configuration Settings for User Self Service
 - 1.1.4.19.1.1 Configuring SAP Business Suite System for User Self Service
 - 1.1.4.19.1.2 Configuring SAP NetWeaver System for User Self Service
 - 1.1.4.19.1.3 User Self Service IMG Activities
 - 1.1.4.19.2 Creating a User Account
 - 1.1.4.19.3 Resetting the Password
 - 1.1.4.19.4 User Self Service Enhancement Options
 - 1.1.4.20 Integration Scenarios
 - 1.1.4.20.1 Integration of SAP Gateway and SPI
 - 1.1.4.20.1.1 Generating an SAP Gateway Service for SPI
 - 1.1.4.20.1.2 Mapping Between SPI and OData Model
 - 1.1.4.20.2 Integration of SAP Gateway and GenIL
 - 1.1.4.20.2.1 Generating an SAP Gateway Service for GenIL
 - 1.1.4.20.2.2 Enhancing the Classes
 - 1.1.4.20.2.3 Mapping Between GenIL and OData Model
 - 1.1.4.20.3 Integration of SAP Gateway and SAP Business Warehouse
 - 1.1.4.20.3.1 Generating an SAP Gateway Analytics Service Using MDX and Easy Query
 - 1.1.4.20.4 SAP Gateway With SAP HANA
 - 1.1.4.20.4.1 Creating and Adding a HANA DB Based SAP Gateway Service

- 1.1.4.20.4.2 SAP Gateway with SAP HANA APIs
 - 1.1.4.20.4.2.1 //WHDB/CL_HAI_RT_ABS_MODEL
 - 1.1.4.20.4.2.2 //WHDB/CL_HAI_RT_DATA
- 1.1.4.20.5 OData Services Consumption and Integration
 - 1.1.4.20.5.1 Creating an RFC Destination
 - 1.1.4.20.5.2 Generating an SAP Gateway Service for OData
 - 1.1.4.20.5.3 Enabling Media Links
- 1.1.4.20.6 Model Composition for Integration Scenarios
 - 1.1.4.21 Subscription and Notification Flow
 - 1.1.4.21.1 Subscription and Notification Flow for Push Oriented Scenarios
 - 1.1.4.21.1.1 Creating a New OData Channel Application for Sending Notifications
 - 1.1.4.21.1.1.1 Configurations on the SAP Backend System for Push Oriented Scenarios
 - 1.1.4.21.1.1.2 Configurations on the SAP Gateway System for Push Oriented Scenarios
 - 1.1.4.21.1.2 Notification Content Publisher
 - 1.1.4.21.1.3 Notification Content Formatter
 - 1.1.4.21.2 Subscription and Notification Flow for Pull Oriented Scenarios
 - 1.1.4.21.2.1 Creating a New OData Channel Application for Pulling Notifications
 - 1.1.4.21.2.1.1 Configurations on the SAP Backend System for Pull Oriented Scenarios
 - 1.1.4.21.2.1.2 Configurations on the SAP Gateway System for Pull Oriented Scenarios
 - 1.1.4.21.2.1.2.1 Pulling the Notifications Stored - Enhancements
 - 1.1.4.21.3 Enabling Push/Pull for an Existing OData Channel Application
 - 1.1.4.21.4 Maintaining Outbound bgRFC Queue from SAP Backend System to the Hub System
 - 1.1.4.21.5 Maintaining Inbound bgRFC Queue on the Hub System
 - 1.1.4.21.6 Subscription Management
 - 1.1.4.21.6.1 An Example to Create Subscription on Behalf of Users
 - 1.1.4.21.6.2 Fetching the List of Users for Roles
 - 1.1.4.21.6.3 MOC for Subscription Management
 - 1.1.4.22 Namespace Handling in Model Provider Class
 - 1.1.4.22.1 //WBEP/IF_MGW_EXPR_VISITOR
 - 1.1.4.23 Integration with Unit Test Framework
- 1.1.5 APIs and Coding
 - 1.1.5.1 Logging in SAP Gateway
 - 1.1.5.1.1 Initiating the Logger
 - 1.1.5.1.2 Completing a Logging Step
 - 1.1.5.1.3 Logging an Exception
 - 1.1.5.1.4 Logging BAPI Returns
 - 1.1.5.1.5 Logging Simple Messages
 - 1.1.5.1.6 Logging Simple Messages Through a Structure
 - 1.1.5.1.7 Logger Handling for Outbound Flows
 - 1.1.5.1.8 Non-User and Request-Specific Logging CCMS
 - 1.1.5.1.9 Message Container
 - 1.1.5.2 OData Channel APIs
 - 1.1.5.2.1 OData Channel Runtime APIs
 - 1.1.5.2.1.1 //WBEP/CL_COS_LOGGER
 - 1.1.5.2.1.2 //WBEP/CL_MGW_ABS_DATA
 - 1.1.5.2.1.3 //WBEP/IF_MESSAGE_CONTAINER
 - 1.1.5.2.1.4 //WBEP/IF_MGW_APPL_SRV_RUNTIME
 - 1.1.5.2.1.5 //WBEP/IF_MGW_CONV_SRV_RUNTIME
 - 1.1.5.2.1.6 //WBEP/IF_MGW_ENTRY_PROVIDER
 - 1.1.5.2.1.7 //WBEP/IF_MGW_ODATA_EXPAND
 - 1.1.5.2.1.8 //WBEP/CX_MGW_BUSI_EXCEPTION
 - 1.1.5.2.1.9 //WBEP/IF_MGW_REQ_FUNC_IMPORT
 - 1.1.5.2.1.10 //WBEP/IF_MGW_REQ_FILTER
 - 1.1.5.2.1.11 //WBEP/IF_MGW_REQ_ENTITY_C
 - 1.1.5.2.1.12 //WBEP/IF_MGW_REQ_ENTITY_U
 - 1.1.5.2.1.13 //WBEP/IF_MGW_REQ_ENTITY_D
 - 1.1.5.2.1.14 //WBEP/IF_MGW_REQ_ENTITY_P
 - 1.1.5.2.1.15 //WBEP/IF_MGW_REQ_ENTITY
 - 1.1.5.2.1.16 //WBEP/IF_MGW_REQ_ENTITYSET
 - 1.1.5.2.1.17 //WBEP/IF_MGW_EXPR_VISITOR
 - 1.1.5.2.2 OData Channel Metadata APIs
 - 1.1.5.2.2.1 //WBEP/CL_MGW_ABS_MODEL

- 1.1.5.2.2.2 //WBEP/IF_MGW_MED_ODATA_TYPES
- 1.1.5.2.2.3 //WBEP/IF_MGW_ODATA_ACTION
- 1.1.5.2.2.4 //WBEP/IF_MGW_ODATA_ANNOTATABLE
- 1.1.5.2.2.5 //WBEP/IF_MGW_ODATA_ANNOTATION
- 1.1.5.2.2.6 //WBEP/IF_MGW_ODATA_ASSOC
- 1.1.5.2.2.7 //WBEP/IF_MGW_ODATA_ASSOC_SET
- 1.1.5.2.2.8 //WBEP/IF_MGW_ODATA_CMPLX_PROP
- 1.1.5.2.2.9 //WBEP/IF_MGW_ODATA_CMPLX_TYPE
- 1.1.5.2.2.10 //WBEP/IF_MGW_ODATA_DOCUMENTTN
- 1.1.5.2.2.11 //WBEP/IF_MGW_ODATA_ENTITY_SET
- 1.1.5.2.2.12 //WBEP/IF_MGW_ODATA_ENTITY_TYP
- 1.1.5.2.2.13 ABAP Dictionary Type to EDM.Type Mapping
- 1.1.5.2.2.14 ABAP Dictionary Type to EDM.Type Mapping SP10
- 1.1.5.2.2.15 //WBEP/IF_MGW_ODATA_ITEM
- 1.1.5.2.2.16 //WBEP/IF_MGW_ODATA_MODEL
- 1.1.5.2.2.17 //WBEP/IF_MGW_ODATA_NAV_PROP
- 1.1.5.2.2.18 //WBEP/IF_MGW_ODATA_PARAMETER
- 1.1.5.2.2.19 //WBEP/IF_MGW_ODATA_PROPERTY
- 1.1.5.2.2.20 //WBEP/IF_MGW_ODATA_REF_CONSTR
- 1.1.5.2.3 OData Vocabulary Annotations APIs
 - 1.1.5.2.3.1 //WBEP/IF_MGW_VOCAN_ANNOTATION
 - 1.1.5.2.3.2 //WBEP/IF_MGW_VOCAN_ANN_TARGET
 - 1.1.5.2.3.3 //WBEP/IF_MGW_VOCAN_COLLECTION
 - 1.1.5.2.3.4 //WBEP/IF_MGW_VOCAN_FUNCTION
 - 1.1.5.2.3.5 //WBEP/IF_MGW_VOCAN_FUN_PARAM
 - 1.1.5.2.3.6 //WBEP/IF_MGW_VOCAN_LABEL_ELEM
 - 1.1.5.2.3.7 //WBEP/IF_MGW_VOCAN_PROPERTY
 - 1.1.5.2.3.8 //WBEP/IF_MGW_VOCAN_RECORD

SAP Gateway Foundation Developer Guide

Use

The **SAP Gateway Foundation Developer Guide** provides information to help you run, develop, and enhance applications and SAP solutions that integrate end user programs with SAP systems in the context of SAP Gateway foundation. This guide covers the following topics:

- [OData Channel](#)
An OData-based minimalistic programming approach.
- [SAP Gateway Service Builder](#)
This is the single entry point for all developers who want to create an OData service with ease.
- [Workflow Services](#)
The OData Channel services provide a Workflow task service for the Business Workflow.
- [SAP Gateway Cookbooks](#)
A few short how-to guides which allow you to get started with the different programming approaches and get some applications up and running quickly.

1.1 OData Channel

Use

In SAP Gateway, entities are defined and connected in data models. These data models can be exposed as OData services during runtime. Runtime processing is handled by data provider classes (DPC) that are assigned to the object models.

The **OData Channel (ODC)** for SAP Gateway allows you to develop content by defining object models and registering a corresponding runtime data provider class. There are two ODC application components, the business data provider and the metadata provider. An ODC data provider implements a single interface that is very close to the OData protocol. OData Channel applications obtain access to common framework services, for example, logging or transaction handling using a set of agents called "Common Service Exposure". These agents wrap existing framework services in a way that OData Channel applications can easily consume.

For SAP NetWeaver systems lower than SAP NetWeaver 7.4, you need to install the software component `IW_BEP` either on the SAP Gateway system or on the SAP Business Suite backend system. For SAP NetWeaver 7.4 systems and higher, the component `SAP_GWEND` is installed as standard and includes the functional scope of `IW_BEP`. For more information about the SAP Gateway components required, see [Deployment Options](#).

Content Programming Model

SAP Gateway enables the content implementation to use OData features. As such, the content inherits the flexibility and complexity of OData.

The provided metadata reflects the implementation of the business data APIs.

OData provides a wide range of request features such as "inlining/expand", batches, paging, sorting, and filtering. The content implementation balances between the flexibility offered by OData and the technical constraints of the business functionality used.

Customizing

SAP Gateway supports multiple SAP backend systems as providers of OData content. Since OData content can be provided by multiple SAP systems, you must configure the connectivity between the SAP backend systems and the SAP Gateway system.

If multiple SAP backend systems are connected to a single SAP Gateway system, the SAP Gateway system resolves potential service name conflicts.

For example: Many customers have local HR systems in an international system landscape. These local systems may contain identical SAP Gateway content that is exposed by one SAP Gateway system. SAP Gateway provides the flexibility to provide specific URLs for the resources derived out of the SAP Gateway content.

For information on the customizing activities which are available for OData Channel, see [OData Channel Configuration](#).

Note

Note that services can only be processed if SAP Gateway has been activated globally.

To activate SAP Gateway, access the SAP Implementation Guide (IMG) in transaction `SPRO` and navigate to ► **SAP NetWeaver** ► **SAP Gateway** ► **OData Channel** ► **Configuration** ► **Activate or Deactivate SAP Gateway** ►

More Information

- [OData in SAP Gateway - Feature Overview](#)
- [General Considerations](#)
- [Basic Features](#)
- [Advanced Features](#)
- [APIs and Coding](#)

OData in SAP Gateway - Feature Overview

Use

Supported OData Features

SAP Gateway supports the following OData features:

OData	Feature	Description	Available as of
-------	---------	-------------	-----------------

Format	ATOM	Atom is an XML-based document format that describes lists of related information known as feeds.	SAP Gateway Release 2.0 Support Package 03
Format	ATOM - literal representation	Literal representation of primitive types must be used in OData URI or HTTP header.	SAP Gateway Release 2.0 Support Package 03
Format	ATOM - Feed customization (FC_TargetPath only)	A service may represent the value of an entry property as the value of a standard Atom element (for example, title, summary).	SAP Gateway Release 2.0 Support Package 03
Format	JSON	JavaScript Object Notation (JSON) is a lightweight data interchange format based on a subset of the JavaScript Programming Language standard.	SAP Gateway Release 2.0 Support Package 04
Format	OpenSearch description	The OpenSearch description document format can be used to describe a search engine so that it can be used by search client applications.	SAP Gateway Release 2.0 Support Package 04
Operations	HTTP Status Code	Control HTTP status code in exceptional case in data provider class.	SAP Gateway Release 2.0 Support Package 05
Operations	CRUD	HTTP Post, Get, Updated and Delete requests on resources.	SAP Gateway Release 2.0 Support Package 03
Operations	Read - Media Resources	Read a binary association to a media link entry. The media link entry can either point to an internal binary in a table, for example, or to an external link that must be set in the data provider.	SAP Gateway Release 2.0 Support Package 03
Operations	CUD - Media Resources	Create, Update, and Delete media resources associated with a media link entry.	SAP Gateway Release 2.0 Support Package 04
Operations	Concurrency Control and eTag (e.g. If-Match, If-None-Match)	HTTP ETags are used for concurrency control.	SAP Gateway Release 2.0 Support Package 03
Operations	Batch Handling	Batch requests allow the grouping of multiple operations into a single HTTP request payload.	SAP Gateway Release 2.0 Support Package 04
Operations	Repeatable Requests / Idempotency	Any kind of operation which can be repeated by providing a request ID. A repeated request is processed by the idempotency framework (fallback), for example, a create action not performed twice.	SAP Gateway Release 2.0 Support Package 05
Operations	Patch	An operation to perform a partial update which contains just a subset of the entry's properties.	SAP Gateway Release 2.0 Support Package 05
Operations	Deep Insert	An operation to create an entity with deep data in an in-lined format, for example, to create a parent/child relationship simultaneously.	SAP Gateway Release 2.0 Support Package 03
Operations	Expand	URI with a \$expand system query option indicates that entries associated with the entry or collection of entries identified by the resource path section of the URI must be represented inline.	SAP Gateway Release 2.0 Support Package 03
Operations	Function Imports with Return Type Entry, Feed ComplexType	Service operations represent functions exposed by an OData service. These functions may accept zero or more primitive type parameters.	SAP Gateway Release 2.0 Support Package 03
Operations	Function Import with Return Type Collection of Complex Types	Function imports with return type collection of complex types.	SAP Gateway Release 2.0 Support Package 03
Operations	Merge / Patch	Update request sent via HTTP method MERGE / PATCH indicates that a subset of the entry's properties is sent to the server and should be merged with those fields that are not present in the request.	SAP Gateway Release 2.0 Support Package 05
Query Options	Paging (\$top, \$skip)	The \$top and \$skip system query option identifies a subset of the entries in the collection of entries.	SAP Gateway Release 2.0 Support Package 03
Query Options	\$filter	The \$filter system query option identifies a subset of the entries in the collection of entries.	SAP Gateway Release 2.0 Support Package 03
Query Options	\$count (handled by ODC - not populated to DPC API)	The \$count system query option specifies that the request response contains a count of the number of entries	SAP Gateway Release 2.0 Support Package 03

		in the collection only.	
Query Options	\$count (populated to DPC API)	The \$count system query option specifies that the request response contains a count of the number of entries in the collection only.	SAP Gateway Release 2.0 Support Package 05
Query Options	\$orderby	The \$orderby system query option specifies an expression for determining what values are used to order the collection of entries.	SAP Gateway Release 2.0 Support Package 03
Query Options	\$select populated to DPC API	The value of \$select specifies that a response from an OData service should return a subset of the properties of the collection of entries.	SAP Gateway Release 2.0 Support Package 04
Query Options	\$select (handled by ODC generically)	The value of \$select specifies that a response from an OData service should return a subset of the properties of the collection of entries.	SAP Gateway Release 2.0 Support Package 03
Query Options	\$links (read only; handled by ODC)	The \$links option specifies that the response to the request contains the links between an entry and an associated entry or a collection of entries.	SAP Gateway Release 2.0 Support Package 03
Query Options	Read property - value, raw value (\$value) and complex property	OData provides two ways of exposing individual properties, one that follows OData formats and another that exposes the property's raw value.	SAP Gateway Release 2.0 Support Package 03
Query Options	\$inlinecount=allpages	A URI with a \$inlinecount system query option specifies that the response to the request includes a count of the number of entries in the collection of entries.	SAP Gateway Release 2.0 Support Package 04
Query Options	\$skiptoken (server-driven paging)	The request response contains an <atom:link rel="next"> element for partial representations of the collection of entries.	SAP Gateway Release 2.0 Support Package 04
SAP-Specific Extensions/Features	Multi Origin / Multi Destination (origin segment parameter)	A service may need to connect to several backend systems. The server adds an additional key to the collection of entries, which indicates the original system. In addition, the server defines strategies to deal with implicit and explicit CREATE requests.	SAP Gateway Release 2.0 Support Package 03
SAP-Specific Extensions/Features	Multi origin/multi destination (parallelization)	A service may need to connect to multiple backend systems and the backend calls are done in parallel.	SAP Gateway Release 2.0 Support Package 05
SAP-Specific Extensions/Features	Versioning (segment parameter)	A segment parameter in the URI of the service, for example, /myservice;v=2 may indicate a request to a specific service version. In addition, the SAP Gateway Catalog Service offers a function import for version negotiation between the client and the server.	SAP Gateway Release 2.0 Support Package 04
SAP-Specific Extensions/Features	Conversion Handling	Conversions done in ABAP are handled implicitly for some conversion routines such as for currencies (based on metadata extracted from the ABAP Data Dictionary) or explicitly by the application developer if required.	SAP Gateway Release 2.0 Support Package 03 Also available for Compatibility Mode for SP02
SAP-Specific Extensions/Features	Error messages - Micro Format	The error response to a request can be influenced by the developer and, if required, a detailed error description can be provided in a micro format. This is only possible for ATOM format.	SAP Gateway Release 2.0 Support Package 03
SAP-Specific Extensions/Features	XSRF Token-based protection mechanism	The protection XSRF attacks is ensured via a token-based exchange mechanism in the HTTP request/response header between the client and the server.	SAP Gateway Release 2.0 Support Package 03

1.1.1.1 OData Query Options

The OData query options supported by SAP Gateway belong to either of these categories:

- They are automatically available in the framework, no further actions needed by development.
- They are available, but they require implementation efforts from development.

Table 1:

OData Query Options	Additional Implementation Needed
\$select	no
\$count	no
\$expand	no
\$format	no
Read \$links	no
\$value	no
\$orderby	yes
\$top	yes
\$skip	yes
\$filter	yes
\$inlinecount	yes
\$skiptoken	yes

Query Options: No Additional Implementation Needed

The following framework query options are available automatically.

- \$select

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$select=Category,Name
```

- \$count

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$count
```

- \$expand

Request example:

```
https://<server>:<port>/.../<service_name>/Suppliers?$expand=Products
```

- \$format

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$format=json
```

- Read \$links

Request example:

```
https://<server>:<port>/.../<service_name>/Products('1000')/$links/Category
```

- \$value

Request example:

```
https://<server>:<port>/.../<service_name>/Products('1000')/Name/$value
```

Query Options Requiring Implementation

The following query options can be used, but they require implementation efforts by development.

- \$orderby

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$orderby=Category desc
```

- \$top

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$top=5
```

- \$skip

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$skip=10&$top=5
```

- \$filter

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$filter=Category eq 'Notebooks'
```

- \$inlinecount

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$inlinecount=allpages
```

- \$skiptoken

Request example:

```
https://<server>:<port>/.../<service_name>/Products?$skiptoken=20
```

Example

Implementation example:

```
METHOD products_get_entityset.
```

```
DATA: lt_return          TYPE TABLE OF bapiret2,
      lt_filter          TYPE /iwbsp/t_mgw_select_option,
      ls_filter          TYPE /iwbsp/s_mgw_select_option,
      lt_category_filter TYPE STANDARD TABLE OF bapi_epm_product_categ_range,
```



```

ls_category_filter TYPE bapi_epm_product_categ_range,
ls_sel_option      TYPE /iwbep/s_cod_select_option,
lv_top             TYPE int4,
lv_skip           TYPE int4,
lv_skiptoken      TYPE int4,
lt_entityset      TYPE zcl_zag_product_mpc=>tt_product,
ls_entity         TYPE zcl_zag_product_mpc=>ts_product,
lt_orderby        TYPE /iwbep/t_mgw_tech_order,
ls_orderby        TYPE /iwbep/s_mgw_tech_order,
lv_count          TYPE i.

* Filter
lt_filter = io_tech_request_context->get_filter( )->get_filter_select_options( ).

LOOP AT lt_filter INTO ls_filter WHERE property EQ 'CATEGORY'.
  LOOP AT ls_filter-select_options INTO ls_sel_option.
    CLEAR ls_category_filter.
    MOVE-CORRESPONDING ls_sel_option TO ls_category_filter.
    APPEND ls_category_filter TO lt_category_filter.
  ENDLOOP.
ENDLOOP.

CALL FUNCTION 'BAPI_EPM_PRODUCT_GET_LIST'
  TABLES
    headerdata      = lt_entityset
    selparamcategories = lt_category_filter
    return          = lt_return.

* Order By
lt_orderby = io_tech_request_context->get_orderby( ).
READ TABLE lt_orderby INTO ls_orderby INDEX 1.
IF sy-subrc EQ 0.
  IF ls_orderby-order EQ 'desc'.
    CASE ls_orderby-property.
      WHEN 'PRODUCT_ID'.
        SORT lt_entityset BY product_id DESCENDING.
      WHEN 'NAME'.
        SORT lt_entityset BY name DESCENDING.
      WHEN 'CATEGORY'.
        SORT lt_entityset BY category DESCENDING.
    ENDCASE.
  ELSEIF ls_orderby-order EQ 'asc'.
    CASE ls_orderby-property.
      WHEN 'PRODUCT_ID'.
        SORT lt_entityset BY product_id ASCENDING.
      WHEN 'NAME'.
        SORT lt_entityset BY name ASCENDING.
      WHEN 'CATEGORY'.
        SORT lt_entityset BY category ASCENDING.
    ENDCASE.
  ENDIF.
ENDIF.

* paging
lv_top = io_tech_request_context->get_top( ).
lv_skip = io_tech_request_context->get_skip( ) + 1.

"Client Paging (top/skip)
IF lv_top IS NOT INITIAL OR
lv_skip NE 1.
  LOOP AT lt_entityset INTO ls_entity FROM lv_skip.
    APPEND ls_entity TO et_entityset.
    ADD 1 TO lv_count.
    IF lv_count = lv_top.
      EXIT.
    ENDIF.
  ENDLOOP.
ELSE.
  "No Paging
  et_entityset = lt_entityset.
ENDIF.

"Server Paging (skiptoken)
lv_skiptoken = io_tech_request_context->get_skiptoken( ).
* for server paging functionality (not implemented in this example)
* you need to return a fixed size of entries (e.g. 100) and fill export parameter
* es_response_context-skiptoken

```

```

* Inlinecount
IF io_tech_request_context->has_inlinecount( ) = abap_true.
    DESCRIBE TABLE et_entityset LINES es_response_context-inlinecount.
ELSE.
    CLEAR es_response_context-inlinecount.
ENDIF.

ENDMETHOD.

```

Related Information

<http://www.odata.org> 

1.1.2 General Considerations

Use

In this section you can find information on the following:

- [OData Best Practices](#)
- [Performance Best Practices](#)
- [Creating High-Quality OData Services](#)
- [Deployment Considerations for Development](#)

1.1.2.1 OData Best Practices

Use

This section contains useful dos and don'ts for developers programming SAP Gateway applications and optimizing performance.

Dos

- **Do think in REST terms**

If you find yourself inventing function imports such as `CreateX`, `ReadX`, `GetX`, `ChangeX`, `UpdateX`, or `DeleteX`, you should make the entity type `X` part of your model.

CRUD is covered by `GET`, `PUT`, and `DELETE` to entities and `POST` to entity sets. .

Function imports may be used for operations other than CRUD, such as `Approve`, `Reject`, `Reserve`, and `Cancel`. However, you might find that a reservation entity that you can create, update, and delete is better suited.

- **Do return completely filled entries in response to `GET ~/EntitySet`**

Think:

```
SELECT * FROM EntitySet WHERE ($filter) ORDER BY ($orderby) UP TO ($stop) ROWS
```

if you see the following:

```
GET ~/EntitySet?$filter=...&$orderby=...&$stop=...
```

OData does not distinguish `GetList` from `GetDetail`. OData clients expect to obtain all information from reading an entity set. If the information is not transparent and comprehensive, OData clients might attempt subsequent updates that could lead to errors in your data.

- **Do use the OData system query options `$filter`, `$top`, `$skip`, `$orderby`**

Clients can, and should, use the query options to tailor responses to your needs. Content creators should support these query options to enable clients to optimize requests.

- **Do provide a usable set of filterable properties**

Make as many properties as possible filterable. At least all primary and foreign key properties should be filterable.

- **Do make your properties concrete**

- **Do use the right data type for your properties**

- **Do represent quantities and monetary amounts as `Edm.Decimal`**

Use the annotation `sap:unit` for the amount or quantity property to point to the corresponding currency or unit. If you need to distinguish currency units from units of measure, use the corresponding `sap:semantics` annotation.

In this way, clients like Microsoft Excel™ can work with the numbers directly and mobile devices, for example, can display them using the device-specific locale setting.

- **Do use `Edm.IntXX` or `Edm.Decimal` for NUMC fields**

This takes care of leading zeros and alpha conversions.

- **Do use media resources and media link entries**

Binary data in common formats (for example, PDF, GIF, JPEG) is naturally represented in OData as a media resource with a corresponding media link entry, that is, a structured record describing the media resource, and containing a link to it. In this way, the binary can be accessed directly via its own URL, just like a browser accesses a picture in the Internet. In fact, a browser can then access your binary data directly, without needing to know OData. Another benefit is performance. Binary data as a media resource is streamed directly byte by byte, whereas binary data hidden within an OData Property of type `Edm.Binary` is represented as a string of hex digits, thereby doubling its size.

- **Do provide navigation properties to link related data**

If your model contains `Customers` and `SalesOrders`, provide navigation between them as `/Customers(4711)/SalesOrders` and `/SalesOrders(42)/Customer`. This will ensure that the client knows how to construct a query on `SalesOrders` using a `CustomerID`, or the URI of the customer resource from the `CustomerID`.

If one of your entity types has an `xxxID` property, make sure you also provide a corresponding `xxx` entity type and navigation to it. The same applies for `xxxCode`; provide an entity set that lists all possible code values and meaningful descriptions for them. The service should be self-describing.

- **Do follow links provided by the server**

URI construction rules can change, but the basic convention surrounding links will not. Following links is therefore future-proof.

1 Note

Strive to develop high-quality OData services. For more information, see [Creating High-Quality OData Services](#).

Don'ts

- **Don't think in SOAP terms**
For more information, see: Do think in REST terms.
- **Don't construct URIs in your client application**
For more information, see: Do follow links provided by the server.
- **Don't invent your custom query options**
Clients cannot discover custom query options for themselves. You have to define how your clients are to use every single custom query option. Use function imports with `HttpMethod="GET"` returning collection of entity type if you need to expose special queries that do not fit `$filter`. Function imports are described in the `$metadata` document and can be discovered by clients.
- **Don't force the client to construct links**
For more information, see: Do provide navigation properties to link related data.
- **Don't use `Placeholder001` or `ForFutureUse` properties.**
For more information, see: Do make your properties concrete.
- **Don't use binary properties**
For more information, see: Do use media resources and media link entries.
- **Don't use generic name-value entity types**
It is inappropriate to express entity-relationship models with just two entities, one named `Entity` and one named `Relationship`.
- **Don't just tunnel homegrown XML**

➔ Recommendation

For more information, read the OData Protocol documentation:

<http://www.odata.org/documentation/overview> 📌

If you are unsure, speak to experienced colleagues in development for advice.

1.1.2.2 Performance Best Practices

This section contains useful dos and don'ts for developers with a special focus on performance aspects.

Dos

- Do use **JSON format** to reduce client and SAP Gateway server time and reduce network traffic.
- Do use GZIP to compress data.

OData query options that optimize performance and end-to-end resource consumption:

- Do use **delta query** in case the end users trigger the same query calls with the same options very often. This will fetch only the data from the SAP Business Suite that was created/changed/deleted since the end-user last asked.
- Do use **\$select** to specify the fields returned in the result set.
- Do use **\$top** and **\$skip** (client-side paging) to reduce the returned items in the result set.
- Do use **\$filter** to fetch a subset of the entities in the entity set.
- Do use **\$skiptoken** (server-side paging) to fetch a subset of the entities in the entity set starting from the first entity at the index identified by the value of the **\$skiptoken** query option.
- Do use **\$count** to fetch only the count of a collection of entities.
- Do use **\$inlinecount** to fetch the count of a collection of entities together with the queried entities.
- Do use **\$expand/deep insert** for associated entity types when you need to fetch or create deep data. Translated into the ODBC-language, **\$expand** is the possibility of OData to make 'joins'. Deep inserts are the possibility to make updates on views.
 - Do consider using parallel calls in case of **one** association if it is beneficial to show the results of the faster call. Use **\$expand/deep insert** for one association call. For parallel calls this option is not possible.
 - Do use **\$expand/deep insert** for more data which contains more than one association call.
 - Do implement and use basic **\$expand**. Basic **\$expand** is a central feature to provide the possibility to the application to read and return entities deeply (entity set and entity). Consider not to use default (generic) **\$expand**.

\$expand performance optimization, in comparison to single calls, includes the following:

- **\$expand** reduces roundtrips from the client to the SAP Gateway server as well as the SAP Gateway server to the SAP Business Suite. As a result, network time is reduced.
- **\$expand** reduces SAP Gateway server response time and the consumption of resources.
- Basic **\$expand** implementation can reduce the SAP Business Suite times, because the number of database operations and the application logic can be reduced.

Do use **\$batch** to send the data of several requests in a single HTTP request from the client to the SAP Gateway server for entity types that are not associated:

- Do use parallel calls in case of two Read calls or two Update calls with **multiple** atomic units of work (different change sets). As an alternative to two parallel calls, you can consider using **\$batch**.
- Do use **\$batch** in case of more than two calls without associated entity types (do not make too many client – SAP Gateway server roundtrips).

\$batch performance optimizations and considerations (compared to single calls):

- **\$batch** reduces the number of roundtrips between the end-user and the SAP Gateway server. As a result, network time is reduced.
- **\$batch** reduces the number of roundtrips from the SAP Gateway server to the SAP Business Suite to one in case of Update calls with a single atomic unit of work.
- **\$batch** reduces SAP Gateway server response time and the consumption of resources.
- **\$batch** reduces CPU time of the SAP Business Suite in case of Update calls with a single atomic unit of work.
- Consider the balance between the number of roundtrips and the payload size. A payload size which is too big can affect the network time due to bandwidth constraints.

- Consider using \$batch because it can have a negative impact on the response time since SAP Gateway calls SAP Business Suite/s in sequential order (in case of Read calls or Update calls with multiple atomic units of work).

The following table can help to choose the required option between \$expand, \$batch and parallel calls.

	Number of Calls from the Client to SAP Gateway	
Operation	Two Calls	More Than Two Calls
Association Read	\$Expand/ parallel calls	\$Expand
Read without Association	Parallel calls	\$Batch
Update with Multiple Atomic Units of Work	Asynchronous (parallelized calls)/\$Batch	\$Batch
Update with Single Atomic Units of Work	\$Batch	

Don'ts

- Do **not** trigger more than two roundtrips from the client to the SAP Gateway server per single operation.
- Do **not** use \$batch for associated entity types. Use \$expand instead.
- Do **not** fetch more data than will be presented. Use OData query options to reduce the result set to fit data to be presented.
- Do **not** call more than two parallel calls per single operation. Use \$batch for more than two parallel calls. More than two parallel calls consume more server resources and network bandwidth.
- Do **not** use generic (default) \$expand functionality provided by the SAP Gateway framework. The generic expand function will not be ideal with regard to performance since it only represents a feature to provide functional completeness automatically. Implement and use basic expand to optimize performance and reduce SAP Gateway server and SAP Business Suite response time and the consumption of resources.

Whether a specific \$expand is better than the generic one depends on the database / data access function design.

RFC Bypass

If a SAP Gateway system and the backend system are located on the same computer, a shortcut should be used to speed up performance during the communication between the SAP Gateway framework on both systems. Using this shortcut, a serialization and deserialization and also a Remote Function Call from SAP Gateway to the backend system is no longer necessary. This speeds up the performance significantly.

Note that this shortcut is **not** available for multiple origin composition (MOC).

To activate or deactivate the shortcut communication an IMG activity is available: In transaction SPRO open the SAP NetWeaver reference IMG and navigate to **SAP Gateway > OData Channel > Configuration > Connection SAP Gateway to SAP System > Manage SAP System Alias**. In the **System Alias** entry customized for your service(s) you can mark or unmark parameter **Local SAP GW** to activate or deactivate the shortcut communication.

1.1.2.3 Creating High-Quality OData Services

Use

To improve the quality of your OData services developed with SAP Gateway, consider the following recommendations, which describe how to improve the service representation from a consumer perspective.

More Information

For more general information about OData guidelines, see [OData Best Practices](#).

Dos and Don'ts

Recommendations for Model Provision (Metadata)

Level	Don't	Do
Schema		Do add a document element (mandatory for function imports).
Entity Type	Don't use technical names or field names of underlying data sources (such as BAPI, function modules).	Do use readable and understandable terms.
	Don't use uppercase or snake case (SALES_CONTACT).	Do use upper camel case (SalesContact).
	Don't use the same name for a property or navigation property and its enclosing entity type.	Do add document element at entity type level.
Property (Naming)	Don't use uppercase or snake case (CUSTOMER_NAME) or a combination of both for property names.	Do use upper camel case (CustomerName).
	Don't use technical names or field names of underlying data sources (such as BAPI, function modules).	Do use readable and understandable terms.
Property (Typing)	Don't use edm:string for everything.	Do use appropriate (specific) property types.
Property (SAP Annotations)	Don't use unit of measure or currency code properties without reference.	Do maintain SAP annotations appropriately. Use sap:label at property level (mandatory). All other annotations must be maintained if they differ from default value.
		Do set SAP semantic to unit of measure or currency code for unit of measure and currency code properties.

		Do ensure that at least one property with the annotation <code>filterable=true</code> exists if the corresponding entity set has the annotation <code>requires-filter=true</code> .
		Do ensure that at least one property should have the annotation <code>filterable=true</code> if the corresponding entity set has the annotation <code>addressable=true</code> .
Property (Feed Customization)		Do set appropriate feed customization for <code>atom:title</code> .
		Do set appropriate feed customization <code>atom:updated</code> .
		Do ensure that the property type does match the target path <code>FC_TargetPath</code> .
		Do ensure the following if the <code>hasstream</code> attribute is set to <code>true</code> and there is a property annotated with <code>FC_TargetPath='content/@src'</code> : <ul style="list-style-type: none"> Property has type <code>string</code> (as it is intended to contain the URI of the corresponding media resource for each entry of this entity type). Exactly one other string property is annotated with <code>FC_TargetPath="content/@type"</code> (indicating that this property contains the mime type of the media resource).
Property (Others)	Don't use large binaries in an OData stream.	Do use media resources and media link entries.
	Don't use a large number of properties.	Do keep the number of properties as low as necessary. Consider using complex types to structure your properties.
		Do validate your terms again dictionary (for example, SAPTerm, Babylon).
Navigation Property (Naming)	Don't use HTTP method name (such as <code>GET</code>) as prefix, suffix or infix.	Do use names that reflect the navigation end points as well as the cardinality.
Association (Naming)		Do use names that reflect the corresponding association cardinality (<code>Product_Supplier_Supplier_Products</code>).
Entity Set (Naming)		Do use plural term of the entity type for entity sets (<code>Products</code>).
Entity Set (SAP Annotations)		Do set the <code>addressable</code> annotation at entity set level to <code>false</code> so user does not query entity set directly.
		Do ensure that at least one property with the annotation <code>requires-filter</code> exists or at least one property without the annotation <code>requires-filter</code> exists if entity set has the annotation <code>requires-filter=true</code> .
Function Import	Don't use HTTP method name (such as <code>GET</code> as prefix, suffix or infix).	Do use <code>ProductsByRating</code> .
		Do use appropriate HTTP method. A <code>GET</code> method must have a return type. Post should have parameters.

Recommendations for Data Provision (Runtime)

Level	Don't	Do
Entry	Don't return error messages that are not user friendly, that is, that are not easily understandable and clear.	Do ensure the <code>src</code> attribute of the content element points to a valid URI if the corresponding entity type has attribute <code>m:HasStream=true</code> .
	Don't use empty values for <code>DateTime</code> such as <code>0000:00:00T00:00:00</code> or <code>T00:00:00</code> since this leads to errors.	Do ensure that all collections with <code>sap:addressable=true</code> have appropriate returns.
		Do ensure the implemented methods do not have side effects. For example, a <code>Get</code> method must not change or create any objects in the system.
		Do use exception class <code>/IWBEP/CX_MGW_BUSI_EXCEPTION</code> for errors related to business logic.
		Do catch exceptions from business logic and, if needed, use your own error messages.
		Do fill <code>DateTime</code> values appropriately.
		Do implement factory pattern based on entity set name to distinguish between requested sets if multiple entity set definitions exist for a single entity type.

Recommendations for OData Channel

If you adopt an outside-in approach (that is you want to base a service on an existing metadata file), we propose you use the Import Model function to generate a basic implementation. Using the Import Model function enables you to use common structures and a predefined programming pattern and style.

Level	Don't	Do
Service Endpoint	Don't use <code>sap-client</code> in the URI of service endpoints.	Do activate the service for the appropriate <code>sap-client</code> .
		Do use <code>Compatibility Mode for SP2</code> , but only if it is explicitly required for your project.
Entity Type Property Complex Type	Don't use data element <code>CHAR30</code> for everything.	Do choose appropriate DDIC elements like data elements or structures for data binding.
Entity Type Complex Type	Don't reuse existing structures with a high number of fields if only a few of them are used in the data provider class.	Do use appropriate DDIC structures with a limited number of properties.
Property		Do maintain the field labels of your data elements appropriately. The medium field label is used as <code>sap:label</code> by default. A meaningful label improves the usability of the consuming UI when used as a column header, for example.

1.1.2.4 Deployment Considerations for Development

Use

Depending on whether you want to develop SAP Gateway content in an SAP Business Suite backend system or in an SAP Gateway hub system, you can choose to implement SAP Gateway in an embedded or central hub deployment scenario.

Deployment Overview

For more information, see:

- [Development in SAP Gateway Hub System](#)
- [Development in SAP Business Suite Backend System](#)

Development in SAP Gateway Hub System

Use

If you do not want to make changes (for example, implement new application coding) in your SAP Business Suite system and remote interfaces exist in the SAP Business Suite system that fulfill your requirements, you can install the relevant SAP Gateway components and develop SAP Gateway content in an SAP Gateway hub system. This is referred to as hub deployment.

If the SAP Gateway hub system in which you want to develop content is based on:

- SAP NetWeaver 7.0 EHP2, 7.03, or 7.31, you need to install the component `IW_BEP`.
- SAP NetWeaver 7.4 or higher, the SAP Gateway core component `SAP_GWFND` is installed as standard (includes the same functional scope as `IW_BEP`) and no further installation is required.

Hub Deployment

Before deciding whether or not hub deployment is the most appropriate development setup for you, consider the following aspects that affect system performance in a remote setup:

- Number of round-trips to exchange data between the systems.
- Volume of data which needs to be transferred between the systems.
- Function scope you want to utilize in a particular system.

In a system landscape with hub deployment, communication between systems is handled by system aliases (RFC destinations). If the remote interface fits your use case and permits the exchange of metadata such as filtering, paging, searching or sorting statements then a development approach with a hub deployment might be the most ideal solution. It is best suited if data can be accessed with a low number of round-trips between the SAP Gateway system and the SAP Business Suite system.

More Information

For more information, see [Development in SAP Business Suite Backend System](#).

1.1.2.4.2 Development in SAP Business Suite Backend System

Use

If you need to develop flexible OData services that serve multiple use cases, it is advantageous to have the application code in the same system in which your SAP Business Suite data resides.

If the SAP Business Suite system in which you want to develop SAP Gateway content is based on:

- SAP NetWeaver 7.0, 7.01, 7.02, 7.03, 7.31, you need to install the component `IW_BEP`.

- SAP NetWeaver 7.4 or higher, the SAP Gateway core component `SAP_GWFND` is installed as standard (includes the same functional scope as `IW_BEP`) and no further installation is required.

Embedded Deployment

SAP Gateway communicates efficiently with systems in which `IW_BEP` or `SAP_GWFND` is installed. These SAP Gateway components transfer all the data to the SAP Business Suite system where the request is processed. Communication between the SAP Gateway system and the SAP Business Suite system is optimized for OData handling so you do not need to develop new remote interfaces for each use case.

SAP Gateway can return both deep (complex) data and flat data. Any metadata such as filtering, paging, and sorting information is passed to the SAP Business Suite system and can be delegated to the underlying database instead of being transferred to the SAP Gateway system. By developing SAP Gateway content directly in the SAP Business Suite system you can maximize the ABAP database capabilities already in place.

More Information

For more information, see [Development in SAP Gateway Hub System](#).

1.1.3 Basic Features

Use

In this section you can find information on the following basic features:

- [Catalog Service](#)
- [Media Links](#)
- [Service Life-Cycle](#)

1.1.3.1 Catalog Service

The catalog service provides a list of all available services for your area. Exploration of OData services is crucial for several tools to identify a suitable service for the realization of a business use case. Therefore certain search capabilities are required.

Metadata exploration is available using standard OData mechanisms. It is built on top of OData Channel to have full OData feature support when accessing the local database. Exposure of metadata uses search capabilities (scanning the English version of a description of a service), for example, search for all service documents where `SalesOrder` is used in the description.

Note

Exploration of the remote OData Channel's services and the registration of its services on SAP Gateway is also provided.

The **catalog service** retrieves a list of all available services on SAP Gateway. It is based on a catalog service pattern proposed by Microsoft™ and consists of an implementation approach of the catalog service pattern in the context of SAP Gateway.

Note

There are two versions of the catalog service available in SAP Gateway framework, Version 1 and Version 2. Version 2 has EntitySet collection, Tag collection, Vocabulary collection and Annotations. The open search functionality is also enhanced in Version 2. You can choose the version based on your requirement.

Activate and Maintain Services									
Service Catalog									
Type	Technical Service Name	Version	Service Description	External Service Name	Nsp.	OAu.	Soft State	Processing Mode	
BEP	C_CFD_TSM_TAX_PUBLISH_CDS	1	TSM Tax Classification	C_CFD_TSM_TAX_PUBLISH_CDS			Not Supported	Routing-based	
	/IWFND/SG_MED_CATALOG	1	Catalog Service	CATALOGSERVICE	/IWFND/	✓	Not Supported	Routing-based	
	/IWFND/SG_MED_CATALOG	2	Catalog Service Version 2	CATALOGSERVICE	/IWFND/			Routing-based	
BEP	CDS_MH_TEST_ODATA_TRANSI	1	SADL Generated Service	CDS_MH_TEST_ODATA_TRANSI			Not Supported	Routing-based	
BEP	CFD_I_TSM_COMPANY_CDS	1	SADL Generated Service	CFD_I_TSM_COMPANY_CDS			Not Supported	Routing-based	

External tools and developers can use this service to retrieve all available OData services, which are exposed on one dedicated SAP NetWeaver system. The catalog service thus acts as a central entry point to access a list of available services.

The following functionality is offered:

- Free text search is now based on the service description, service ID, service name.
- Filtering which means that all attributes can be set to filterable.
- Support of the [delta token](#) functionality
- EntitySet Collection: The EntitySet collection lists all the entity sets that are available in different services. This cannot be created / deleted / updated from the client.
- Tag Collection: The Tag collection lists all the backend tags that are associated with various services. These tags cannot be deleted via catalog service. For example,
`http://<server>:<port>/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/ServiceCollection('%2FIWFND%2FSG_MED_CATALOG_0002')/Tags`. Here, `%2FIWFND%2FSG_MED_CATALOG_0002` is the associated service.
- Open search also covers EntitySet collection and Tag collection while searching. For example, if you do an open search with `FlightCollection` it is expected to search for `RMTSAMPLEFLIGHT` service.

The catalog service is available via the following URL:

`http://<server>:<port>/sap/opu/odata/iwfnd/CATALOGSERVICE/`

The following pictures show an example catalog service, first the service document, then the service metadata document.

```
<?xml version="1.0" encoding="utf-8" ?>
- <app:service xml:lang="en" xml:base="https://[redacted]/sap/opu/odata/iwfind/CATALOGSERVICE/"
  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:searchable="true" sap:content-version="1"
    href="ServiceCollection">
    <atom:title type="text">ServiceCollection</atom:title>
    <sap:member-title>Service</sap:member-title>
    <atom:link href="ServiceCollection/OpenSearchDescription.xml" rel="search" type="application/opensearchdescription+xml"
      title="searchServiceCollection" />
    </app:collection>
  - <app:collection sap:content-version="1" href="CatalogCollection">
    <atom:title type="text">CatalogCollection</atom:title>
    <sap:member-title>Catalog</sap:member-title>
    </app:collection>
  </app:workspace>
  <atom:link rel="self" href="https://[redacted]/sap/opu/odata/iwfind/CATALOGSERVICE/" />
  <atom:link rel="latest-version" href="https://[redacted]/sap/opu/odata/iwfind/CATALOGSERVICE?v=2/" />
</app:service>
```

```
<?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:DataService m:DataServiceVersion="2.0">
- <Schema Namespace="CATALOGSERVICE" xml:lang="en" xmlns="http://schemas.microsoft.com/ado/2008/09/edmx">
- <EntityType Name="Service" sap:content-version="1">
  - <Key>
    <PropertyRef Name="ID" />
  </Key>
  <Property Name="Description" Type="Edm.String" Nullable="false" MaxLength="60" m:FC_TargetPath="SyndicationTitle" m:FC_KeepInContent="true" sap:label="Description" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
  <Property Name="Title" Type="Edm.String" Nullable="false" MaxLength="40" sap:label="External Name" sap:creatable="false" sap:updatable="false" />
  <Property Name="Author" Type="Edm.String" Nullable="false" MaxLength="12" m:FC_TargetPath="SyndicationAuthorName" m:FC_KeepInContent="true" sap:label="User Name" sap:creatable="false" sap:updatable="false" />
  <Property Name="TechnicalServiceVersion" Type="Edm.Int16" Nullable="false" sap:label="Technical Service Version" sap:creatable="false" />
  <Property Name="ID" Type="Edm.String" Nullable="false" MaxLength="40" sap:label="Identifier" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
  <Property Name="Metadataurl" Type="Edm.String" Nullable="false" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
  <Property Name="TechnicalServiceName" Type="Edm.String" Nullable="false" MaxLength="35" sap:label="Technical Service Name" sap:creatable="false" />
  <Property Name="Imageurl" Type="Edm.String" Nullable="false" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
  <Property Name="Serviceurl" Type="Edm.String" Nullable="false" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
  <Property Name="UpdatedDate" Type="Edm.DateTime" Nullable="false" Precision="7" m:FC_TargetPath="SyndicationUpdated" m:FC_KeepInContent="true" sap:label="Time Stamp" sap:creatable="false" sap:updatable="false" sap:filterable="false" />
</EntityType>
- <EntityType Name="Catalog" sap:content-version="1">
  - <Key>
    <PropertyRef Name="ID" />
  </Key>
  <Property Name="Uri" Type="Edm.String" Nullable="false" sap:creatable="false" sap:filterable="false" />
  <Property Name="UpdatedDate" Type="Edm.DateTime" Nullable="false" Precision="7" sap:creatable="false" sap:filterable="false" />
  <Property Name="Imageurl" Type="Edm.String" Nullable="false" sap:creatable="false" sap:filterable="false" />
  <Property Name="ID" Type="Edm.String" Nullable="false" sap:creatable="false" sap:filterable="false" />
  <Property Name="Description" Type="Edm.String" Nullable="false" sap:creatable="false" sap:filterable="false" />
  <Property Name="Title" Type="Edm.String" Nullable="false" sap:creatable="false" sap:filterable="false" />
  <NavigationProperty Name="Services" Relationship="CATALOGSERVICE.Services" FromRole="FromRole_Services" ToRole="ToRole_Services" />
</EntityType>
- <Association Name="Services" sap:content-version="1">
  <End Type="CATALOGSERVICE.Catalog" Multiplicity="1" Role="FromRole_Services" />
  <End Type="CATALOGSERVICE.Service" Multiplicity="*" Role="ToRole_Services" />
</Association>
- <EntityType Name="ServiceCollection" EntityTypeName="CATALOGSERVICE.Catalog" sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="1">
  <EntitySetName="CatalogCollection" EntityTypeName="CATALOGSERVICE.Catalog" sap:content-version="1" />
  <AssociationSet Name="AssocSet_Services" Association="CATALOGSERVICE.Services" sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:content-version="1">
    <End EntitySet="CatalogCollection" Role="FromRole_Services" />
    <End EntitySet="ServiceCollection" Role="ToRole_Services" />
  </AssociationSet>
- <FunctionImport Name="BestMatchingService" ReturnType="CATALOGSERVICE.Service" EntitySetName="ServiceCollection" m:HttpMethod="GET">
  <Parameter Name="TechnicalServiceVersionMin" Type="Edm.Int16" Mode="In" Nullable="false" />
  <Parameter Name="TechnicalServiceName" Type="Edm.String" Mode="In" Nullable="false" />
  <Parameter Name="TechnicalServiceVersionMax" Type="Edm.Int16" Mode="In" Nullable="false" />
</FunctionImport>
</EntityType>
- <EntityContainer>
  <atom:link rel="self" href="https://[redacted]/sap/opu/odata/iwfind/CATALOGSERVICE/$metadata" xmlns:atom="http://www.w3.org/2005/Atom" />
  <atom:link rel="latest-version" href="https://[redacted]/sap/opu/odata/iwfind/CATALOGSERVICE?v=2/$metadata" xmlns:atom="http://www.w3.org/2005/Atom" />
</EntityContainer>
- <Schema>
  <edmx:DataServices>
  </edmx:DataServices>
</edmx:Edmx>
```

Mapping of System ID to System Alias via HTTP Response Header

For some applications the service calls are made using the SID: this is the ABAP system ID. This does not correspond to the SAP Gateway system alias. To allow these applications to also use standard requests via the SAP Gateway system alias, you need a mapping from SID to system alias. The service catalog has been enhanced by a function import which retrieves a list of system aliases for both client and system name.

Example: `GetSystemAliasesForSID?CLIENT='999'&SYSTEM='ABC'`

An error is thrown if the client is wrong. An empty response is displayed if both client and system are ok but there is no result as to the system alias.

1.1.3.2 Media Links

Use

You can use and define statically defined and read-only resources (referenced by a stable URL) within SAP Gateway. This enables applications to expose information from different media types, such as graphics and videos, for example.

The OData Channel provides a generic solution for exposing binary data that is stored in an SAP Business Suite backend system and should be accessible using a media link entry.

- Binaries can be exchanged separately between the consumer, SAP Gateway system, and SAP Business Suite backend system.
- The application can set the respective HTTP headers via an API, for example, for cache control for the respective media resource (for non-sensitive data only).

Prerequisites

- **Media link representation:**

An entity type of an SAP Gateway data model that is to be treated as a media type has to be defined as such in its properties. For such entity types, you must select the **Media** checkbox in its properties. Based on the media type classification, the OData Service Metadata Document sets the `hasStream` to `true` for the resulting `EntityType`.

Node structure: An entity type that is marked as a media link has to have the following properties to support the correct handling of the media link:

- **Source URI**

External resources: this is used by the consumer to retrieve the associated media resource. For internal resources, the resource manager can be

used in the application logic at runtime to set the media resource URI properly. In a read scenario of a static media resource, the Source URI contains a static link to the resource. The property which contains the source URI is annotated by the application developer by using method `SET_AS_CONTENT_SOURCE`.

Note

OData Channel only:

Binary resources, for example stored in the SAP Business Suite backend system: the source URI of the media link entry is automatically generated by the framework. When accessing the link, the framework routes the request to the read stream implementation of the data provider class. Note that this is only required if you want to point to a media resource which refers to an external URL. If you handle the resource call with your own implementation in `GET_STREAM`, then you do not have to set this property reference at all.

- MIME type

This describes the media resource's type, for example `image/png`. The mime type of the media resource is set by the application logic at runtime. The property which contains the source URI is annotated by the application developer by using method `SET_AS_CONTENT_TYPE`. The content type always has to be set.

Implementation Considerations for OData Channel

To use this function to retrieve a raw value, you must include an implementation for each service in the SAP Business Suite backend system. The entity type has to be marked as `HasStream` and the method `Get_Stream` has to be implemented.

The `/ $value` is an additional command to an entry to identify the raw value of an entry. For example, if an entry represents an engine with the additional `/ $value`, an image of the engine can be retrieved.

Example

The following is an example of the source code for the software component `IW_BEP` for the redefined method `DEFINE` of class `/IWBEF/CL_MGW_ABS_MODEL` and `/IWBEF/IF_MGW_APPL_SRV_RUNTIME~GET_STREAM` of class `/IWBEF/CL_MGW_ABS_MODEL`.

Redefined method `DEFINE` of class `/IWBEF/CL_MGW_ABS_MODEL`:

```
* setting a data object as media type means that it gets a special semantic by having a url and all
lo_data_object->set_is_media( ).

lo_property = lo_data_object->create_property(
    iv_property_name = 'mimeType'
    iv_abap_fieldname = 'MIME_TYPE'
).
* must be set when data object is a media type to mark the property which represents the mime type information
lo_property->set_as_content_type( ).

|||

/IWBEF/IF_MGW_APPL_SRV_RUNTIME~GET_STREAM of class /IWBEF/CL_MGW_ABS_MODEL:

    DATA: ls_stream          TYPE          ty_s_media_resource
    ,lo_api          TYPE REF TO  if_mr_api
    .

    lo_api = cl_mime_repository_api=>get_api( i_prefix = '/SAP/PUBLIC/BC/Pictograms' ).
    lo_api->get( EXPORTING i_url          = '3_people_money.gif'
                IMPORTING e_content     = ls_stream-value
                e_mime_type = ls_stream-mime_type ).

    copy_data_to_ref( EXPORTING is_data = ls_stream
                     CHANGING cr_data = er_stream ).
```

1.1.3.3 Service Life-Cycle

A list of the topics within the Service Life-Cycle section.

Use

The service life-cycle encompasses everything in the life span of an OData service, from the activation and maintaining of the service, via the maintaining of models and services, up to the cleanup of the metadata cache. We recommend you use the Service Builder for your applications, but a purely programmatic approach is also possible.

- [SAP Gateway Service Builder](#) is the recommended tool for all aspects of service development.
- [Activate and Maintain Services](#) is the central transaction for service activation. It is also accessible via the Service Builder.
- [Maintaining Models and Services](#)
- [Maintain Annotation Models](#)
- [Metadata Cache](#)
- [Soft State Based Query Result Cache](#)

1.1.3.3.1 Activate and Maintain Services

Describes the actions available within the `Activate and Maintain Services` activity in the IMG.

Use

The transaction for activating and maintaining services is used to maintain all registered services on the SAP Gateway server (hub system), to register and activate services, to delete services, and to simplify the usage in general.

The service maintenance offers the following advantages:

- Fast overview of all registered services on the SAP Gateway hub system as well as backend systems
- Detailed display of a service, such as ICF nodes and system alias assignment
- Easy generation and activation of new services

The service maintenance is part of the Implementation Guide (IMG) in your SAP Gateway system. In the SAP Reference IMG (transaction `SPRO`) navigate to **SAP NetWeaver > SAP Gateway > OData Channel > Administration > General Settings > Activate and Maintain Services**.

Type	Technical Service Name	Version	Service Description	External Service Name	Namespace	SAP
BEP	ZOCCA_C20	1	Cost Centers: Costs and Allocations (Replicated)	OCCA_C20		
	/IWFND/ACCOUNT_CSCS	1	12345323423	ACCOUNT_CSCS	/IWFND/	
	/IWFND/ACCOUNTSCS2_SD	1	dwqwd	ACCOUNTSCS2_SD_test	/IWFND/	
	/IWFND/SG_EPM_ADDRTYPE	1	F4 Help values for AddressType	ADDRESSTYPE	/IWFND/	
BEP	ZASW_FLIGHT	1	ASW	ASW_FLIGHT		
BEP	ZATFIRST_SRV	1	ZCL_ATFIRST_DPC_EXT	ATFIRST_SRV		
	ZUSER_BANK_DEMO	1	Bank Demo using BOR Generator	BANKDEMOANIR		
	ZBANKREFCDEMO	1	Demo Bank using RFC generator	BANKREFCANIR		
	Z_BANK_TEST_UC	1	Bank Test	BANKTESTRFC		
	/IWFND/SG_EPM_BPROLE	1	F4 Help values for BP Role	BPROLE	/IWFND/	
	Z_SG_EPM_BP	1	test for BP	BUSINESSPARTNER		

The main screen is divided into an upper and a lower part in which the upper part shows all registered services (**Service Catalog**) and the lower part shows the details of the selected service from the **Service Catalog** . The details are split into **ICF Nodes** and **System Aliases** .

In the **Activate and Maintain Services** screen you can access some basic SAP Gateway functions directly:

- You can quickly access the SAP Gateway Client by choosing **Goto > SAP GW Client** . The SAP Gateway Client provides general testing functions for your OData services.
- You can quickly access the metadata cache by choosing **Goto > Cleanup of Model Cache** .

Service Catalog

The service catalog lists all services of the current system. You can easily find services from a backend system that have not yet been registered by choosing **Add Service** in the menu bar.

The list of services that will be added is shown as soon as a system alias is provided. The list can be further limited by entering additional search criteria as filter values.

Type	Technical Service Name	Version	Service Description	External Service Name	Namespace	External Mapping ID
BEP	/IPGW/METAR_O2C_CONTEXT	1	PGW Process Context Service: Order to Cash Test P.	METAR_O2C_CONTEXT	/IPGW/	
BEP	/IPGW/SLS_ORDR_PRCSNG_ORDR_1	1	PGW Process Context Service: Sales Order Processin.	SLS_ORDR_PRCSNG_ORDR_1	/IPGW/	
BEP	/IPGW/SLS_ORDR_PRCSNG_ORDR_2	1	PGW Process Context Service: Sales Order Processin.	SLS_ORDR_PRCSNG_ORDR_2	/IPGW/	
BEP	/IPGW/SLS_ORDR_PRCSNG_ORDR_I	1	PGW Process Context Service: Sales Order Processin.	SLS_ORDR_PRCSNG_ORDR_I	/IPGW/	

You can select a service from the list under **Add Service** via the hotspot of column **Technical Service Name** , **Service Description** , or **External Service Name** .

Details about the selected service are displayed. In this detailed view you always enter a valid package name or, for local objects which should not be transported, you choose **Local Object** which will automatically assign package `$TMP` to the service.

Service	
Technical Service Name	/IWBEP/Z_SG_NI
Service Version	1
Description	Test
External Service Name	Z_SG_NI
Namespace	/IWBEP/
External Mapping ID	
External Data Source Type	C
Model	
Technical Model Name	Z_SG_NI

Model Version: 1

Creation Information

Package: Local Object

ICF Node

☒ Standard Mode ☐ None

☐ Compatibility Mode for SP 02

☒ Set current client as default client in ICF Node

In section **ICF Node** the option **Set current client as default client in ICF Node** is pre-selected as well as the ICF node to be used. ICF node creation is disabled if the service already has an ICF node. This is because different versions are using the same ICF node / ICF service. If ICF services for both modes (standard as well as compatibility) exist, then both radio buttons would be greyed out. If only one ICF service exists, then only one ICF button will be greyed out. For example, if an ICF node for the compatibility mode already exists, then this option is greyed out and **None** is set. In this case, you could activate the standard mode.

Note

For new development the **standard mode** is always used. For more information, see [ICF Services](#).

Service versioning helps an administrator to generate a service with an additional version. In addition, detailed information of each service can be displayed to get a first idea on the different versions. Thus an administrator can easily handle different versions of a service that are provided by the backend system(s).

Apart from activating single services, you can also activate several services at once. This **mass activation** can be useful for large amounts of services that have to be available quickly. Of course you need several services to start with. To carry out mass activation, proceed as follows:

1. Choose **Add Service**.
2. Enter the relevant system alias. You can use a filter, if needed.
3. In the list, select those services that you want to activate and choose **Add Selected Services**.
4. Enter a prefix for your service and model names, for example Z and enter a valid package and choose **Enter**.

A list of all the services that were created is displayed, together with a status indicator (green for successful execution). Now you can close this screen.

The list of selected backend services no longer contains those services that you have activated.

For the **Service Catalog**, the following main functions are available:

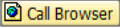
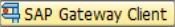
Button	Description
Filter	An administrator can use this to search for a service with the technical service name, the service version, the external service name, etc.
Add Service	Used to add a service (see above).
Delete Service	Used to delete the selected service. Caution If you want to delete a service completely, all ICF nodes and system alias assignments have to be deleted manually. With the introduction of version handling, the deletion of services has also been enhanced: As long as more than one version per service exists, the ICF nodes do not have to be deleted.
Service Details	Displays additional information about the corresponding model(s) for a service, such as the model name, the version, the external mapping ID, and the model description.
Load Metadata	Reloads the service metadata for the annotation models. See Maintain Annotation Models
Error Log	Calls the Error Log for the selected service.
Refresh Catalog	Refreshes all the entries in the Service Catalog.
OAuth	The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. By enabling the service for OAuth the ICF-Handler is changed. For more information about OAuth, see Enabling OAuth 2.0 .
Soft State	Activates or de-activates soft state for a single service. For more information, see Soft State Support .

ICF Nodes

If the ICF node of your service is being activated the traffic light in front of it switches to green.

For the ICF nodes the following functions are available:

Button	Description
ICF Node	Allows you to carry out functions related to the ICF nodes. You can do the following: <ul style="list-style-type: none"> • Create and/or activate the selected ICF node in the list of ICF nodes.

	<ul style="list-style-type: none"> Deactivate a selected active ICF node in the list of ICF nodes. The traffic light switches to yellow. Delete a selected active or inactive ICF node in the list of ICF nodes. The traffic switches to grey (initial). Configure the ICF node. This function calls the general transaction for Internet Communication Framework service maintenance (transaction <code>SICF</code>).
 Call Browser	<p>Calls the default browser and shows the selected service.</p> <p>The browser is called with the segment parameter for versions, for example, <code>.../ServiceDocument;v=0001/...</code></p>
 SAP Gateway Client	<p>Calls the SAP Gateway Client for further analysis.</p> <p>If you have selected a service, this will take you directly to the relevant information for the current service. From there you can proceed further to service implementation details, for example.</p>

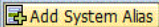
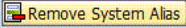
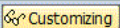
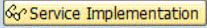
Service Activation Results

Activation of an ICF node yields the following main results:

- Service repository object
Every service which is exposed on the SAP Gateway system has a corresponding repository object. This repository object entry is used for the following:
 - Handling authorizations on service level which can be controlled via setting the authorization object `S_SERVICE`.
 - External name mapping to internal technical definition.
 - Routing to the backend and assignment of system alias.
 - Transportation of an activated service, for example from a development or test system to a productive system.
- Object group
Every service repository object is mapped to an object group which is used for the following:
 - System alias assignment to the backend system. As a default the object group is assigned to the system alias from which it has been activated.
- ICF node
Every service has a corresponding ICF node which has the following purposes:
 - Fine granular settings of authentication mechanism for each service.
 - Activation/deactivation of a service.

System Aliases

For the system alias assignment, the following functions are available:

Button	Description
 Add System Alias	<p>Calls the standard customizing activity Assigning SAP System Aliases to OData Service for the selected service from the service catalog to add another system alias assignment.</p> <p>When selecting a system alias, all other service information, such as <code>Technical Service Name</code>, will be added automatically.</p>
 Remove System Alias	<p>Deletes the selected SAP system from the list of system aliases.</p>
 Customizing	<p>Switches to the maintenance of the assignment of SAP system aliases to OData services.</p>
 Service Implementation	<p>This function allows the forward navigation to the backend system where the service is registered. By double-clicking on the data provider class there you arrive at the class implementation directly.</p>

Example

You can find an example of service maintenance in [2.1 Activate the Service](#).

More Information

[SAP Gateway Client](#)

[Metadata Cache](#)

[Error Log](#)

1.1.3.3.2 Maintaining Models and Services

OData Channel implementations retrieve the data from an SAP Business Suite system, that is, a backend system. For this you first have to define a service in the backend system. Models and services have to be registered.

You can maintain these settings in IMG activities. In transaction `SPRO` open the SAP Reference IMG and navigate to [SAP NetWeaver](#) > [SAP Gateway Service Enablement](#) > [Backend OData Channel](#) > [Service Development for Backend OData Channel](#) > [Maintain Models](#) and [Maintain Services](#).

In these activities you can create your own service by specifying the model name and the model provider class (MPC). In the second activity you specify the service name and the corresponding data provider class (DPC), and then you can assign the model. For more information, see the documentation of the IMG activities.

Extension for Service

In transaction `SPRO` open the SAP Reference IMG and navigate to [SAP NetWeaver](#) > [SAP Gateway Service Enablement](#) > [Backend OData Channel](#) > [Service Development for Backend OData Channel](#) > [Maintain Services](#).

You can register for a service that it extends an existing service. OData requests for the original, extended service will then be processed by the new, extending service. So while the endpoint (URL) remains the same the original service is replaced with another service extending it.

When changing a service an additional section **Extension for Service** is available with the fields **Technical Service Name** and **Service Version**.



Example

- Service B extends service A.
- Service A has been activated on the SAP Gateway hub as OData service ZA.
- A client calls the OData service ZA.
- The runtime checks if for the requested service (A) extensions exist and again if for the extensions additional extensions etc. exist.
 - If no extensions exist, the request would be delegated to Service A.
 - If extensions exist, the runtime determines the requested extension, in this case Service B.
 - The request is delegated to service B.

Multiple Extensions

It is possible to register more than one service to extend the same service.

If more than one service extends a service the following algorithm is used to identify the "right" extending service:

- First the runtime checks if there are extensions that have not been provided by SAP (flag `IS_SAP_SERVICE`).
 - If yes, the runtime considers only those. That is, the customer or partner extensions win over SAP extensions.
 - Else, the runtime considers all extensions, namely all SAP-delivered extensions.
- Among the found extensions pick the one with the newest CREATED timestamp. In other words, the newest extension wins.

BAdI

The runtime also provides a BAdI to overwrite the default runtime determination process. This overwriting is possible via enhancement spot `/IWBEP/ES_MGW_MED_SER_EXT`.

The algorithm described above is the default implementation of the BAdI (class `/IWBEP/CL_MGW_MED_SER_EXT_BADI`).

1.1.3.3.3 Maintain Annotation Models

Describes how to register the annotation provider class with one or more services.

You can register vocabulary-based annotation models. This entails the registration of the annotation provider class (stand-alone annotation files are defined in an annotation provider class) with one or more services.

Annotation Model Name	Annot.	Annotation Model Provider Class	Description	Changed by	Changed at
/IWBEP/TEA_TEST_ANNOTATION_FILE	1	/IWBEP/CL_TEA_VOCAN_PROVIDER	Test Application TEA Annotation file. No semantics here.		08.03.2013 11:02:58
ZHTRAN VOCAN_TEST	1	/IWBEP/CL_TEA_VOCAN_PROVIDER	han's test		05.07.2013 10:32:44

Service Name	Service...	Main Service	Service Namespace Alias
/IWBEP/RMTSAMPLELIGHT	1	<input type="checkbox"/>	
/IWBEP/TEA_TEST_APPLICATION	1	<input checked="" type="checkbox"/>	TEA

You can create, change and delete annotation models. Then you add an assignment, remove an existing assignment and, if you assign more than one service to an annotation provider class, you can classify the most important service as main service: If more than once service per stand-alone annotation file needs to be supported, the annotation registration will allow to flag one service as "leading" service for the annotation file. This service will be used to inherit the SAP Gateway to backend routing.

Annotation provider classes can be registered on the backend via the annotation maintenance, thus creating an annotation repository object. For registering an annotation provider class with one or more services an IMG activity is available in the backend system: In transaction SPRO open the SAP Reference IMG and navigate to **SAP NetWeaver > SAP Gateway Service Enablement > Backend OData Channel > Service Development for Backend OData Channel > Maintain Annotation Models > .**

An annotation provider class is transported and shipped together with the annotation repository object.

After having registered the vocabulary-based annotation file and having linked it to a service you need to do the following:

- Clear the cache in the backend system of that service. See the Metadata Cache section.
- Re-load the metadata of the service to which you have assigned your annotation file. This is needed to make the changes available. For this you switch to the **Activate and Maintain Service** transaction and choose **Load Metadata**. For more information, see the **Activate and Maintain Services** topic. The annotation files are available in the catalog service once the service the annotation file references has been activated on the SAP Gateway system.
- Call the catalog service's annotation feed to see the new annotation file.

Then the annotation runtime is visible via the corresponding catalog service URLs.

Stand-alone annotation files can be discovered via the annotation file collection of the catalog service

(`.../sap/opu/odata/IWFND/CATALOGSERVICE/AnnotationFileCollection`). This collection contains all annotation files where the service they reference is active on the SAP Gateway hub system. An annotation file entity of this collection will have properties, such as:

- Annotation file name
- Annotation file description
- Annotation file media link (which returns the actual annotation file XML)

Support of In-Place Vocabulary Annotations

In-place annotations are defined in a model provider class (MPC) via a method like `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ANNOTATION`.

- Model provider classes are registered on the backend with the service registration, creating a service repository object.
- The model provider class is transported and shipped together with the service repository object.
- The model provider classes are available in the metadata document of a service once the service has been activated in the SAP Gateway hub system.

Prerequisites

You need a valid transport request to transport your settings, unless you are working with local objects.

Related Information

[Metadata Cache](#)

[Activate and Maintain Services](#)

1.1.3.3.4 Metadata Cache

Use

The SAP Gateway metadata component allows a full caching of the metadata which significantly increases the performance of a request which is sent through SAP Gateway.

Three different access scenarios are possible:

- It is required to satisfy a request to the OData service document or the OData service metadata document.
- The SAP Gateway runtime itself needs to access the metadata in order to process a request.
- The `IW_BEP` software component needs to access the metadata in order to process a request.

SAP Gateway delivers functionality for all these scenarios by providing a three-level caching strategy.

Access Scenario 1 - Web Infrastructure Cache

In this scenario consumer requests exist to get the service document or the service metadata document. This means there are requests to resources which change only rarely in productive environments.

In order to use HTTP standard techniques SAP Gateway OData Channel sets the HTTP response header for service (metadata) documents according to the HTTP caching standards (Last-Modified). This parameter enables the web infrastructure components (for example, a web server and browser) to already satisfy requests out of their caches if the resources have already been requested before.

The application-specific model provider classes can influence this time stamp via implementing method `GET_LAST_MODIFIED`. The default implementation of this method derives the last modified time stamp based on the changed time stamp of the model provider class (MPC). The application can overwrite the logic if it does not fit.

Access Scenario 2 - SAP Gateway Metadata Cache

If the request cannot be satisfied via the Web infrastructure cache or if the SAP Gateway runtime itself needs to access the metadata, for example for a read of a feed it is checked whether the metadata of the current service is already present in the SAP Gateway metadata cache (if enabled).

If it is present the data is directly retrieved from the cache. If, on the other hand, it is not present the data is read from the respective data sources and passed back. An example of a data source is model provider class or the SAP Gateway metadata persistency, stored in the SAP Gateway metadata cache (to serve subsequent requests out of it).

Note

In the SAP Gateway OData Channel programming paradigm the model provider class including the last modified time stamp is only called once when initially loading the metadata and storing it in the SAP Gateway metadata cache. If the model provider class is changed afterwards (for example, on account of coding changes or due to an import of a changed model provider class) the SAP Gateway metadata cache performs a handshake for every service document or service metadata document request and checks whether the cache contains most up-to-date metadata. If the metadata is outdated a cache refresh is triggered automatically.

Note that this handshake is **not** performed for business data requests. The cache handshake is only performed in the OData Channel programming model via `IW_BEP` APIs. If the `LAST_MODIFIED` implementation of the model provider always returns the latest timestamp the cache is refreshed in every service document and service metadata document call.

Access Scenario 3- IW_BEP Metadata Cache

The cache on the `IW_BEP` cannot be disabled and requires a proper implementation of model provider class method to retrieve the last modified timestamp.

If the model provider class is not changed but the underlying ABAP Dictionary binding, for example a definition of a field type in an ABAP structure, the changes are not reflected directly as there is no hook for ABAP Dictionary changes. In this case the metadata cache on the `IW_BEP` needs to be cleared manually.

Caching in the Development vs. Productive Landscape

The SAP Gateway metadata cache can be enabled and disabled and is per default deactivated in non-productive systems. We recommend the following cache settings:

• Development systems

The SAP Gateway metadata cache should be disabled in order to always get the latest metadata (default setting).

OData Channel applications should set the latest timestamp in the `GET_LAST_MODIFIED` method of their model provider classes or stick to the default implementation.

• Productive systems

The SAP Gateway metadata cache should always be enabled in order to increase the performance (default setting).

Note

You need to clean-up the metadata cache after every import of a changed model provider class.

- Performance test systems

The SAP Gateway metadata cache should always be enabled in order to increase the performance (default setting).

Performance tests should only be done after the SAP Gateway metadata cache has been fully initialized, that is, at least the service has to have been called once.

Controlling the Metadata Cache

You can make settings for the SAP Gateway metadata cache in the SAP Customizing IMG (implementation guide) in transaction `SPRO`.

In the SAP Gateway hub system open the SAP Reference IMG and navigate to ► **SAP NetWeaver** ► **SAP Gateway** ► **OData Channel** ► **Administration** ► **Cache Settings** ► **Metadata Cache** .

In the `IW_BEP` system open the SAP Reference IMG and navigate to ► **SAP NetWeaver** ► **SAP Gateway Service Enablement** ► **Backend OData Channel** ► **Support Utilities** ► **Clear Cache** .

These activities are available:

- On the SAP Gateway hub system
 - Activate or Deactivate Cache
This activity shows the current status of the cache and allows you to activate/deactivate the metadata cache.
 - Clear Cache
This activity allows you to fully clear the cache or to clear the cache for a range of models.
- On the backend system:
 - Clear Cache
This activity allows you to fully clear the cache or to clear the cache for a range of models.

1.1.4 Advanced Features

The list of the advanced features available for the SAP Gateway developer.

Use

In this section you can find information about the following advanced features:

- [Service Tagging](#)
- [ETag Handling](#)
- [Batch Processing](#)
- [Patch Support](#)
- [Delta Query Support](#)
- [Conversion Handling in OData Channel](#)
- [Expand in Framework and Data Provider](#)
- [Deep Insert](#)
- [Error Response Control for Backend Data Provider](#)
- [\\$filter System Query Option APIs](#)
- [Map Message Container to Message Protocol Format](#)
- [Annotations](#)
- [Extended Support of Long Texts in the Metadata](#)
- [Soft State Support for OData Services](#)
- [Search and OpenSearch](#)
- [Excel Support](#)
- [User Self Service](#)
- [Integration Scenarios](#)
- [Subscription and Notification Flow](#)
- [Namespace Handling in Model Provider Class](#)
- [Integration with Unit Test Framework](#)

1.1.4.1 Service Tagging

Service tagging enables consumption developers to easily find and explore the proper service for their use case. Both tagging and search capabilities are provided. Service developers can thus provide additional information to their service in the catalog service via so-called service tags. This is done with an API in the model provider class (`/IWBEF/IF_MGW_ODATA_MODEL~CREATE_TAG`). Additionally, the entity set names are automatically included in the catalog service (as service tags). Developers can also cluster or categorize a set of services to provide a kind of grouping information (for example, workflow services). In addition, developers can also specify tags in a hierarchical way, for example a category tag with more precise information as sub-tags. Tags can be specified based on a predefined list of common service tags. Administrators, on the other hand, can add additional tags to the ones already specified by the service developer without explicitly touching the model provider class. Administrators can also transport newly added tags without modifications of the existing service.

The following functions and features are offered:

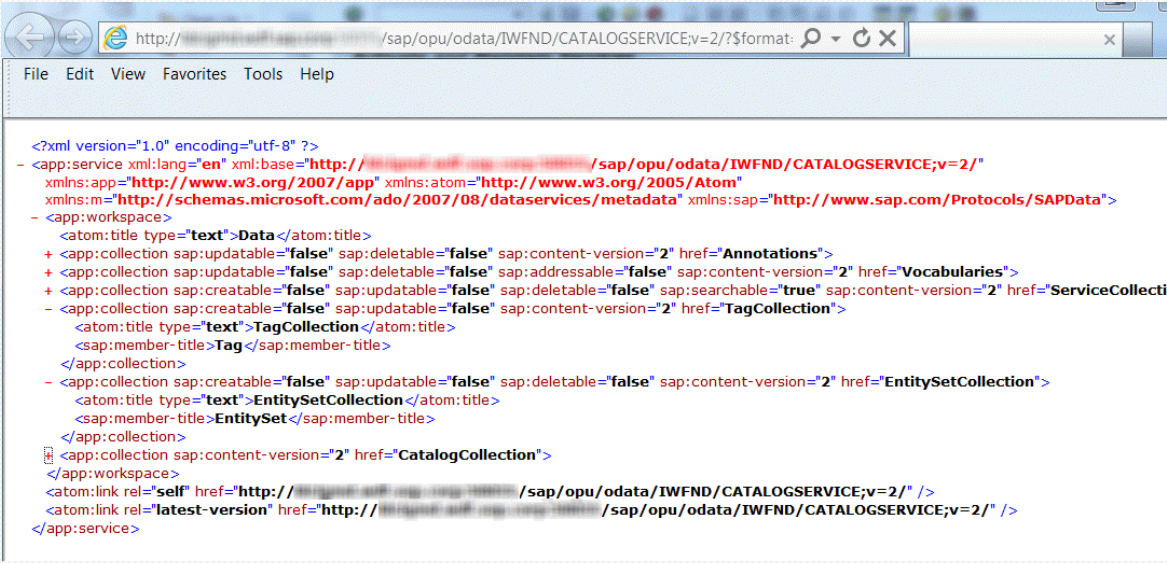
- Tags are defined via the model provider APIs for models.
- Tags are shown as new entities in the service catalog.
- Tags cannot be re-used, that is, there is no concept for tags across models or services. To have the same tags in several services, a developer must make sure to use the same text when creating the tag.
- The open search for services in the service catalog considers the following:
 - Service names
 - Services' entity set external names
 - Catalog service
- A new entity type `Tag` is added. It shows the tags defined with the model provider APIs.

- A new entity type `EntitySet` is added. It shows the entity sets defined with the model provider APIs.

Example

Method `/IWBEP/IF_MGW_ODATA_MODEL~CREATE_TAG`.

`/sap/opu/odata/IWFND/CATALOGSERVICE;v=2` with `TagCollection` and `EntitySetCollection`



```
<?xml version="1.0" encoding="utf-8" ?>
- <app:service xml:lang="en" xml:base="http://[redacted]/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/"
  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:updatable="false" sap:deletable="false" sap:content-version="2" href="Annotations">
  + <app:collection sap:updatable="false" sap:deletable="false" sap:addressable="false" sap:content-version="2" href="Vocabularies">
  + <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:searchable="true" sap:content-version="2" href="ServiceCollecti
- <app:collection sap:creatable="false" sap:updatable="false" sap:content-version="2" href="TagCollection">
  <atom:title type="text">TagCollection</atom:title>
  <sap:member-title>Tag</sap:member-title>
  </app:collection>
- <app:collection sap:creatable="false" sap:updatable="false" sap:deletable="false" sap:content-version="2" href="EntitySetCollection">
  <atom:title type="text">EntitySetCollection</atom:title>
  <sap:member-title>EntitySet</sap:member-title>
  </app:collection>
  <app:collection sap:content-version="2" href="CatalogCollection">
  </app:workspace>
  <atom:link rel="self" href="http://[redacted]/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/" />
  <atom:link rel="latest-version" href="http://[redacted]/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/" />
</app:service>
```

(Catalog service example)

In the catalog service you can search for services, based on Tags or EntitySets by using `$filter`:

`/sap/opu/odata/Iwfnd/catalogservice;v=2/EntitySetCollection?$filter=substringof('Type', Description)`

1.1.4.2 ETag Handling

ETag handling in SAP Gateway.

SAP Gateway offers generic support of ETag (conditions) handling. ETags or entity tags are part of the HTTP protocol. They are header fields that help to determine changes of a resource and can be used in caching scenarios so that redundant data transfer can be minimized. ETag handling is one of several mechanisms that HTTP thus provides for Web cache validation and which allows a client to make conditional requests. OData uses HTTP ETags for optimistic concurrency control.

ETag support in SAP Gateway is provided in both the SAP Gateway hub system and in the SAP Business Suite backend system. Consequently, both hub and backend applications can enhance or modify the generic check. Backend operations can also override the generic conditional handling for data modification requests, such as PUT, MERGE and DELETE in the backend system.

A few special considerations apply for ETags:

- When retrieving an entry, the system returns an opaque ETag value.
- When getting several entries in a feed, the ETag value is included as metadata in the entry itself.
- When retrieving a single entry, the ETag is returned as a response header called ETag as defined by HTTP. The server can choose to also include it in the body as they would do for feeds for consistency.
- During processing of POST, PUT, and MERGE the system should compute a new ETag and return it in a response header, regardless of whether the response has a body with the actual entry information.
- When issuing a PUT, MERGE or DELETE request, clients need to indicate an ETag in the If-Match HTTP request header. If it is acceptable for a given client to overwrite any version of the entry in the server, then the value `***` may be used instead. If a given entry has an ETag and a client attempts to modify or delete the entry without an If-Match header servers should fail the request with a 412 response code.

OData servers will often use weak ETags as a way of indicating that two resources may be semantically equivalent but a particular request may see a different representation of it.

The following interfaces provide methods for ETag handling:

- `IWBEP/IF_MGW_APPL_SRV_RUNTIME` method `GET_IS_CONDITIONAL_IMPLEMENTED`
- `IWBEP/IF_MGW_REQ_ENTITY_U` method `GET_CONDITIONAL_INFO`
- `IWBEP/IF_MGW_REQ_ENTITY_D` method `GET_CONDITIONAL_INFO`

Related Information

[ETag Handling in Batch Processing](#)

1.1.4.2.1 ETag Handling in Batch Processing

A changeset might contain multiple `CREATE/UPDATE/DELETE` operations. This can be problematic in combination with eTags because one operation can change data that is about to be processed by consecutive operations. In this case eTags of these operations might not be valid any more.

However, the application developer has the possibility to use the **defer mode** for changesets by redefining method

`/IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESSET_BEGIN` in the respective DPC class. A data provider can check the entire changeset operations in `IT_OPERATION_INFO` and return `CV_DEFER_MODE = 'X'` to indicate that it does not process the entire changeset operations immediately, but only saves these operations until method `CHANGESSET_PROCESS` is called. A data provider can dynamically decide to process the current changeset in defer mode based on the entire operation info described in table `IT_OPERATION_INFO`. In defer mode, when `CREATE, UPDATE, DELETE` or `EXECUTE ACTION` is called the

data provider has only to save all changeset operations in its internal tables without returning any data or eTag or headers to the framework.
When using defer mode the automatic eTag check will be done for all operations **before** any operations have been processed.

Prerequisites

We recommend the following when using batch in combination with eTags:

1. In your DPC class, redefine method `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESSET_BEGIN` and set parameter `CV_DEFER_MODE` to `ABAP_TRUE` (at least for all relevant operations).
2. If you have redefined method `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_IS_CONDITIONAL_IMPLEMENTED` in your DPC class then make sure that parameter `RV_CONDITIONAL_ACTIVE` is set to `ABAP_FALSE` (at least for all relevant operations). Else you will have to implement the conditional handling for the eTag yourself.

OData

Example coding for using the operation update entity

Code Syntax

Example coding for `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESSET_BEGIN`

```
METHOD /iwbeb/if_mgw_appl_srv_runtime~changeset_begin.  
  
    DATA: ls_operation_info TYPE /iwbeb/s_mgw_operation_info.  
  
    *   Defer Mode only for update entity operations  
  
    cv_defer_mode = abap_true.  
  
    LOOP AT it_operation_info INTO ls_operation_info.  
  
        IF ls_operation_info-operation_type NE /iwbeb/if_mgw_appl_types=>gcs_operation_type-update_entity.  
  
            cv_defer_mode = abap_false.  
  
            EXIT.  
  
        ENDIF.  
  
    ENDLOOP.  
  
ENDMETHOD.
```

Code Syntax

Example coding for `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_IS_CONDITIONAL_IMPLEMENTED`

```
METHOD /iwbeb/if_mgw_appl_srv_runtime~get_is_conditional_implemented.  
  
    IF iv_operation_type EQ /iwbeb/if_mgw_appl_types=>gcs_operation_type-update_entity.  
  
        rv_conditional_active = abap_false.  
  
    ELSE.  
  
        rv_conditional_active = abap_true.  
  
    ENDIF.  
  
ENDMETHOD.
```

Related Information

[/IWBEP/IF_MGW_APPL_SRV_RUNTIME](#)


[\\$batch Processing](#)


[ETag Handling](#)


1.1.4.3 \$batch Processing

Clients can search for and change resources exposed by a data service in a way that each request type maps to a single HTTP request/response exchange. As these request types are based on HTTP, the usual HTTP services are also available, such as caching. In this context clients of a data service can collect or "batch up" several requests and then send that batch to the data service in a single HTTP request.

Batch Request

An OData batch request is represented as a multipart MIME v1.0 message (see RFC2046 under <http://www.rfc-editor.org/rfc/rfc2046.txt> ) , a standard format allowing the representation of multiple parts, each of which may have a different content type, within a single request.

Batch requests allow grouping of multiple operations (as described in OData: Operations under <http://www.odata.org/documentation/operations> ) into a single HTTP request payload.

Batch requests are submitted as a single HTTP POST request to the `$batch` endpoint of a service as described in OData: URI Conventions under <http://www.odata.org/documentation/uri-conventions> .

The body of a batch request is made up of an ordered series of retrieve operations and/or change sets. A change set is an atomic unit of work that is made up of an unordered group of one or more of the insert, update or delete operations. Change sets cannot contain retrieve requests and cannot be nested, that is, a change set cannot contain a change set.

In the batch request body, each retrieve request and change set is represented as a distinct MIME part and is separated by the boundary marker defined in the Content-Type header of the request. The contents of the MIME part which represents a change set is itself a multipart MIME document with one part for each operation that makes up the change set.

Each MIME part representing a retrieve request or change set within the batch includes both Content-Type and Content-Transfer-Encoding MIME headers as seen in the examples below. The batch request boundary is the name specified in the Content-Type Header for the batch.

For more information, see <http://www.odata.org/documentation/operations> .

Batch Response

The batch response contains a Content-Type header specifying a content type of multipart/mixed and a batch boundary specification, which may be different from the batch boundary that was used in the corresponding request.

Within the body of the batch response is a response for each retrieve request and change set that was in the associated batch request. The order of responses in the response body must match the order of requests in the batch request. Each response includes a Content-Type header with a value of `application/http`, and a Content-Transfer-Encoding MIME header with a value of `binary`.

A response to a retrieve request is formatted exactly as it would have appeared outside of a batch.

The body of a change set response is either a response for all the successfully processed change request within the change set, formatted exactly as it would have appeared outside of a batch, or a single response indicating a failure of the entire change set.

For more information, see <http://www.odata.org/documentation/operations> .

Handling Operations in Batch Requests

Custom header (`$batch` response header) is supported for `$batch` requests.

A selection of generic HTTP headers for `$batch` requests are available for the data provider when processing each `$batch` operation.

Performance improvement for `$batch` change-set processing when the data providers is able to handle the entire operation of change set at once at end of change set: For this to be possible the data provider must implement the change set handling API to process all change set operations within one API (`CHANGESET_PROCESS`) and return the consolidated result of all operations to the SAP Gateway framework.

This handling is also called processing a change set in defer mode.

When method `CHANGESET_BEGIN` is called a data provider can use the changing parameter `CV_DEFER_MODE` to inform the framework that it can process all change set operations at once (deferred processing). Based on the list of entity set name, entity type name and action name, a data provider can dynamically set the exporting parameter mentioned above to inform the framework that it will process the current change set at once or to reset this parameter to have a single processing as usual. Default implementation is single processing. That means without any changes in a data provider each change set operation will be processed one after another as usual.

If `CV_DEFER_MODE` is set, the framework will call the data provider using the new method `CHANGESET_PROCESS` with importing parameter `IT_CHANGESET_REQUEST` containing a list of change set operations. Each entry of this list contains the technical request context `IO_TECH_REQUEST_CONTEXT` as usual but also a message container for error or information message happened during the processing. Response data of a change set operation including HTTP custom headers and ETag (if it exists) must be returned in changing parameter `CT_CHANGESET_RESPONSE`.

At the end of a change set the framework will call method `CHANGESET_END` as usual.

All retrieve operations and change sets of `$batch` requests are transferred at once from the SAP Gateway hub system to the backend system for processing.

All consecutive retrieve operations (until a change set) are processed in parallel to improve performance. You can use the implementation guide (IMG) to activate or deactivate the parallelization. See also [Define Parallelization of Batch Queries](#).

Each retrieve operation such as Read Entry or Read Feed within a `$batch` request will be transferred separately from the SAP Gateway hub system to the provider application in the backend system for processing.

Every change set is treated as one Logical Unit of Work (LUW), ensuring its "all or nothing" character. All operations of a change set will be sent at once from the SAP Gateway hub system to the provider application in the backend system for processing.

Results of all operations will be collected at the SAP Gateway hub system and sent as one HTTP response to the OData consumer.

Content ID Referencing

If you have several operations in a change set one operation can refer to another operation by using **content ID referencing**, rather than using the key of an entity type instance which may not be known at that time. To define a content ID for a specific operation the following syntax needs to be used in the `$batch` request body: `Content-ID: n` where `n` is a string selected by the application.

To define a reference to a content ID for a specific operation the following syntax needs to be used in the `$batch` request body: `$n` where `n` is an existing content ID that was defined for another operation from the same change set.


Content ID referencing is supported only for change sets at once (defer mode). As described above in case of "change set at once" method `CHANGESET_PROCESS` is called with importing parameter `IT_CHANGESET_REQUEST` containing a list of change set operations. The importing parameter `IT_CHANGESET_REQUEST` also contains information about content IDs and content ID references (fields `content_id` and `content_id_ref`).

Example

There is a specific business scenario with sales orders, and each sales order can have multiple sales order items. When creating sales orders a sales order ID is generated by the backend. When creating sales order items a sales order id needs to be provided. You want to create a change set with two operations:

- Operation: create sales order
- Operation: create sales order item

This demo business scenario has been implemented in demo service GWSAMPLE_BASIC and can be tested using the following URL and payload:

 **Sample Code**

```
https://<host>:<port>/sap/opu/odata/IWBEP/GWSAMPLE_BASIC/$batch
--batch_005056A5-09B1-1ED1-BF82-409B26A80300
Content-Type: multipart/mixed; boundary=changeset_005056A5-09B1-1ED1-BF82-409B26A80301

--changeset_005056A5-09B1-1ED1-BF82-409B26A80301
Content-Type: application/http
Content-Transfer-Encoding: binary
Content-ID: 100

POST SalesOrderSet HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 1021
<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
<atom:content type="application/xml">
<m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.m
<d:SalesOrderID></d:SalesOrderID>
<d:Note>Deliver as fast as possible</d:Note>
<d:NoteLanguage>EN</d:NoteLanguage>
<d:CustomerID>0100000000</d:CustomerID>
<d:CustomerName>SAP</d:CustomerName>
<d:CurrencyCode>EUR</d:CurrencyCode>
<d:GrossAmount>25867.03</d:GrossAmount>
<d:NetAmount>21737.00</d:NetAmount>
<d:TaxAmount>4130.03</d:TaxAmount>
<d:LifecycleStatus>N</d:LifecycleStatus>
<d:LifecycleStatusDescription>New</d:LifecycleStatusDescription>
<d:BillingStatus/>
<d:BillingStatusDescription>Initial</d:BillingStatusDescription>
<d:DeliveryStatus/>
<d:DeliveryStatusDescription>Initial</d:DeliveryStatusDescription>
<d:CreatedAt>2015-03-22T23:00:00.0000000</d:CreatedAt>
<d:ChangedAt>2015-03-22T23:00:00.0000000</d:ChangedAt>
</m:properties>
</atom:content>
</atom:entry>

--changeset_005056A5-09B1-1ED1-BF82-409B26A80301
Content-Type: application/http
Content-Transfer-Encoding: binary

POST $100/ToLineItems HTTP/1.1
Content-Type: application/atom+xml
Content-Length: 1021

<atom:entry xmlns:atom="http://www.w3.org/2005/Atom">
<atom:content type="application/xml">
<m:properties xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.m
<d:SalesOrderID></d:SalesOrderID>
<d:ItemPosition></d:ItemPosition>
<d:ProductID>HT-1001</d:ProductID>
<d:Note>EPM DG: SO ID 0500000000 Item 0000000020</d:Note>
<d:NoteLanguage>EN</d:NoteLanguage>
<d:CurrencyCode>EUR</d:CurrencyCode><d:GrossAmount>2972.62</d:GrossAmount>
<d:NetAmount>2498.00</d:NetAmount>
<d:TaxAmount>474.62</d:TaxAmount>
<d:DeliveryDate>2015-03-29T22:00:00.0000000</d:DeliveryDate>
<d:Quantity>2</d:Quantity>
<d:QuantityUnit>EA</d:QuantityUnit>
</m:properties>
</atom:content>
</atom:entry>

--changeset_005056A5-09B1-1ED1-BF82-409B26A80301--

--batch_005056A5-09B1-1ED1-BF82-409B26A80300--
```

Additional Application API for \$batch

For retrieve operations there is no special handling required. But to ensure the "all or nothing" character of a change set there are two additional methods for change set handling in interface `/IWBEP/IF_MGW_APPL_SRV_RUNTIME`:

- `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESET_BEGIN`

All operations within a change set must be treated as a logical unit of work. This means all or nothing. Therefore, a provider must **not** issue `COMMIT WORK` or `ROLLBACK WORK` during change set processing. Otherwise, the framework will abandon the change set processing. If the change set contains only one operation, the check of commit or rollback is deactivated.

At the beginning of a change set processing, the provider will be called with this method. It can check the list of all involved entity types and actions contained in this change set and accept the change set handling or reject it by raising a technical exception with exception code `CHANGESET_NOT_SUPPORTED`. The method has a default implementation which allows only one operation per change set in order to guarantee transactional consistency. If more than one operation is required in a change set the implementation has to be overwritten by the application. The application then needs to ensure transactional consistency, for example not to have any commit or rollback in the chain of requests in a change set. Additionally, the provider can start collecting the content of a change set in the modifying method calls, for example update entity, and finalize it within the `CHANGESET_END` method call described below.

- `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~CHANGESET_END`

If the provider only updates all modifications of a change set in internal tables, it can now update the database. A `COMMIT WORK` will be issued by the framework at the end of this API.

More Information

For more information on the API see the method documentation for `CHANGESET_BEGIN` and `CHANGESET_END` of `/IWBEP/IF_MGW_APPL_SRV_RUNTIME`.

For more information on OData operations, see <http://www.odata.org/documentation/operations> .

1.1.4.4 Patch Support

Use

Some update requests support two types of update: replace and merge. The client chooses which to execute by the HTTP verb it sends in the request.

A PUT request indicates a replacement update. The service must replace all property values with those specified in the request body. Missing properties must be set to their default values. Missing dynamic properties must be removed or set to NULL.

A patch or merge request indicates a different update. The service must replace exactly those property values that are specified in the request body. Missing properties, including dynamic properties, must not be altered.

Patch support is provided with method `PATCH_ENTITY` of interface `/IWBEP/IF_MGW_APPL_SRV_RUNTIME`.

1.1.4.5 Delta Query Support

Delta query means "give me all services that were created/changed/deleted since I last asked". In other words, the delta query protocol defines a pull-model protocol for clients to obtain changes from a store. Clients can poll the store for changes periodically or as required, or capable stores can provide change notifications to alert the client when changes are available to be pulled. The underlying design focuses on the ability of the store not to have to keep any per-client state. This delta query function can be used for the [Catalog Service](#), for example.

The delta query is especially useful for projects where performance is critical and loading each time the full payload is not acceptable, for example for projects where the client often polls to get the latest data and any updates. Thus, for client and server performance it can be very useful to always show the current state in a poll-interaction but with the lowest overhead on both client and server

The delta query protocol has the following components:

- **Delta token**

The store generates a token that can be used to retrieve deltas from the current state of the store. This may be returned as part of a request for data or through some other store-defined means.

- **Issuing a request with a delta token**

When the delta token is used to request data, changed or inserted results are returned ordered according to when the changes were made (older changes first).

For changed records, the result must include all changed fields and may contain additional unchanged fields from the record.

Deleted Entries

Deleted entries are a part of delta request (`GET FEED`) where the service returns not only the feed of entities but also the list of entities (to be more exact, entity IDs, called tombstones) that have been deleted since the time point specified in the delta token. The delta token URI parameter is processed and passed in exactly the same way as any other URI parameter, that is, as part of the request context. To support this feature, the backend response XML has been extended with the `<DELETED_ENTITIES>` tag that contains the list of deleted entities. The structure is the same as `<ENTITY_SET>` which is already present to contain the main feed, or a subset of it at least containing all key fields. Only the IDs need to be filled by the backend application.

Note

Note that in the context of JSON deleted entries are **not** supported.

Implementation Considerations

A delta token needs to be issued in a normal feed to signal to the client that it supports delta tokens.

The way the delta token is issued and constructed is done by the issuer and is of his choice. Typically it will be a date-time-token (for default cases). In case of a date-time-token you only need to take into account that the delta token should be related to one specific time zone, such as UTC.

A delta token call from a client needs to be handled appropriately. The delta token holds specific information which is known by the issuer. With this information it should be easy to extract the changed data.

Offlining Considerations

You can implement delta token in several ways and these options can be grouped into two main approaches.

With both approaches, the payload of the response is reduced, but only the first approach is also able to optimize the performance in the backend system.

- The first approach calculates deltas at modification time: The ABAP system tracks relevant changes when they occur. At request time, the deltas are already prepared and thus available. On the one hand, this approach requires more development. On the other hand it is more scalable and has an optimized overall performance.
- The second approach is based on delta determination at request time where the system compares old and new state to find out which records have been changed/deleted. The implementation effort is rather small but it does not optimize the performance of the backend. That means, the more records you have in the full collection, the longer the response time of the request.
For this second approach a basic implementation is described in an SAP Community Network document under <http://scn.sap.com/docs/DOC-47043> with a delta request log component. The implementation effort described is rather low, but the performance on the backend is not optimized.

Examples

Example of issuing a date-time based delta token:

```
* determine delta token for catalog service.
GET TIME STAMP FIELD lv_timestamp.
convert time STAMP lv_timestamp TIME ZONE 'UTC'
      INTO DATE lv_date
      TIME lv_time.
CONCATENATE lv_date lv_time into es_response_context-deltatoken.
```

Example of handling a delta token:

```
data lo_delta_context type /IWBE/IF_MGW_REQ_ENTITYSTDLT
* first get the delta token
lo_delta_context =? io_tech_request_context.
lv_delta_token = lo_delta_context->get_deltatoken( ). "converts it automatically into a datetime
loop at ET_SERVICE_MODEL_INFOS REFERENCE INTO lr_svc_mdl_info.
    TRY .
        if lr_svc_mdl_info->*-last_modified le lv_delta_token.
            * then add it to the return table
            delete ET_SERVICE_MODEL_INFOS index sy-tabix.
        endif.
    * catch as much as possible
    CATCH cx_root INTO lx_exception.
    * service does not exist anymore or is not compileable --> put it to the deleted entries
        lr_svc_mdl_info->*-deleted = abap_true.
    ENDTRY.
endloop.
```

1.1.4.6 Conversion Handling in OData Channel

Use

OData Channel offers a way for applications to delegate handling of conversion exits, currency, currency amount, unit of measurement, and unit amount conversions to the SAP Gateway framework.

Caution

Note that conversion handling functionality is only available in the OData Channel framework that is part of the SAP Business Suite backend system where `IW_BE` is deployed, **not** on the SAP Gateway hub system. If your system is based on SAP NetWeaver 7.4 or higher, you do not need to install component `IW_BE` since the SAP Gateway Foundation component `SAP_GWFND` is installed as standard and includes the functionality of `IW_BE`.

Conversions for the following operations in inbound and outbound flow are available:

- Create entity
- Update entity
- Read entity
- Read entity set

Caution

At present, there is **no** support for conversion of URL parameters.

For example, for a Read operation the Id fields with a conversion exit have to be converted by application coding. If you have an outbound conversion of Id 0100000000 to 1000000000 and try out the input conversion by executing `/sap/opu/odata/sap/my_test/Id('1000000000')` then the method `GetEntity` can not find the instance, because the Id is 1000000000 instead of 0100000000. The self link you get in the feed then has the external representation `/sap/opu/odata/sap/my_test/Id('1000000000')`. For this the application has to provide conversion handling.

The following types conversions are covered:

- Conversions exits (for example, ALPHA)

- Conversion of currency/unit code between SAP internal and external (ISO) ones
- Conversion of currency amounts (for example, a currency amount stored internally as 100.00 may de-facto mean 10.000 if it is about Japanese Yen)
- Conversion of unit amounts

A conversion exit for currency, currency amount, unit of measurement, and unit amount is supported. The model provider class must call the extended method `SET_CONVERSION_EXIT` with the necessary parameters. If no conversion exit is registered, the SAP Gateway framework uses the IDOC function modules `CURRENCY_AMOUNT_SAP_TO_IDOC` and `CURRENCY_AMOUNT_IDOC_TO_SAP` for amount conversion. However, these function modules only work correctly for amounts with two decimal places. For currency and unit of measure, the SAP Gateway framework always uses the global data type conversions (methods for outbound and inbound flow of `CL_GDT_CONVERSION`). You can use method `DISABLE_CONVERSION` of the current property object to disable the property conversion that is otherwise performed automatically by the SAP Gateway framework. You can also use method `ENABLE_CONVERSION` to enable the conversion at property level. Be aware that you can only enable the conversion if method `SET_NO_CONVERSION` is not used for the entire model. Once set, the conversion for the affected property is valid for inbound and outbound flow. The conversion of a property must be disabled in the following cases:

- An amount does not have exactly two decimal places. In this case, your data provider is responsible for converting this amount.
- The OData consumer wants to work with SAP unit of measure or currency and not with ISO codes.

API

In the model provider class the programming interfaces are available for the application to specify that an automatic call of conversion routines is desired. Either a direct API or an indirect API can be used.

Direct API

For each entity field the following methods are available in the `/IWBEP/IF_MGW_ODATA_PROPERTY` interface:

- `SET_CONVERSION_EXIT`
Sets the name of conversion exit to be called for the property.
- `SET_SEMANTIC`
Marks the property as currency code or unit of measurement field.
- `SET_UNIT`
Sets the relation between the amount field and the field containing currency or unit code.
- `ENABLE_CONVERSION`
Enables the conversion for the property.
- `DISABLE_CONVERSION`
Disables the conversion for the property.

Indirect API

Alternatively, applications can use method `BIND_STRUCTURE` either for an entity type or for a complex type (`/IWBEP/IF_MGW_ODATA_CMLX_TYPE`) to tell the framework to take over the conversion information from the corresponding ABAP Dictionary structure. The optional parameter `IV_BIND_CONVERSIONS` has to be set to X.

More Information

For general information on conversion functionality see Input and Output Conversions.

1.1.4.7 Expand in Framework and Data Provider

Describes the expand feature which provides the possibility to the application to read and return entities deeply.

A request with a `$expand` query option enables the reading of entries of an entity together with an associated entity. For example, if you want to read a Sales Order (therefore an entity) and all its associated Items you can execute one request using the `$expand` query option. .



Example

```
https://<server>:<port>/.../<service_name>/SalesOrders('0500000000')?$expand=SalesOrderItems
```

You can also run a `$expand` statement to retrieve multiple Sales Orders (therefore an entity set) including related SalesOrderItems.



Example

```
https://<server>:<port>/.../<service_name>/SalesOrders?$expand=SalesOrderItems
```

OData Channel APIs

The following application interfaces are provided.

- `/IWBEP/IF_MGW_APPL_SRV_RUNTIME`
This interface provides the following methods:
 - `GET_EXPANDED_ENTITYSET`
 - `GET_EXPANDED_ENTITY`
- `/IWBEP/IF_MGW_ODATA_EXPAND`
This interface provides method `COMPARE_TO`.

Enhancements of `/IWBEP/CL_MGW_ABS_DATA`

- Implementation of `GET_EXPANDED_ENTITYSET`
- Implementation of `GET_EXPANDED_ENTITY`

Expand in Framework and Data Provider

Framework Expand

By default SAP Gateway provides a generic `$expand` implementation, which requires no implementation effort. The generic `$expand` statement is handled by the framework and was formerly called "generic expand". When looking at the example above of reading a Sales Orders entity set including related SalesOrderItems entity set, the framework will do the following:

1. It calls the `GET_ENTITYSET` implementation for Sales Orders.
2. For each Sales Order retrieved by the `GET_ENTITYSET` implementation, the framework calls the related `GET_ENTITYSET` implementation for the items in a LOOP.

However, there is an alternative to the framework `$expand`.

Data Provider Expand

The data provider `$expand` which used to be called "basic expand" is based on the data provider and requires implementation effort. The following methods need to be redefined in the related DPC class to enable the data provider `$expand` functionality:

- For reading an entity with an associated entity set (possible use case: you want to read a Sales Order and its associated Sales Order Items):
`/IWBEP/IF_MGW_APPL_SRV_RUNTIME-> /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITY`
- For reading an entity set together with an associated entity set (possible use case: you want to read multiple Sales Orders and all associated Sales Order Items):
`/IWBEP/IF_MGW_APPL_SRV_RUNTIME-> /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITYSET`

Once the methods above are redefined, the data provider `$expand` is active instead of the framework `$expand`.

Performance Considerations

In some cases the implementation of data provider `$expand` can lead to a better performance than the framework `$expand`, depending on the implementation itself and especially on how the data from the business application is fetched. Consider the following example of a data provider `$expand` implementation in method `/IWBEP/IF_MGW_APPL_SRV_RUNTIME-> /IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_EXPANDED_ENTITYSET`.

Example

```
METHOD /iwbeep/if_mgw_appl_srv_runtime~get_expanded_entity.
[...]
```

CALL FUNCTION 'BAPI_EPM_SO_GET_LIST'	
TABLES	
selparamsoid = lt_selparamsoid	
soheaderdata = lt_soheaderdata	
soitemdata = lt_soitemdata.	

```
[...]
ENDMETHOD.
```

The framework `$expand` example that would achieve the same looks different. Since there is no specific framework `$expand` implementation, it is necessary to implement the related `GET_ENTITY/GET_ENTITYSET` methods.

Example

```
METHOD salesorders_get_entity.

[...]
```

CALL FUNCTION 'BAPI_EPM_SO_GET_DETAIL'	
EXPORTING	
so_id = ls_so_id	
IMPORTING	
headerdata = ls_soheaderdata.	

```
[...]

ENDMETHOD.

METHOD salesorderitems_get_entityset.

[...]
```

CALL FUNCTION 'BAPI_EPM_SO_GET_LIST'	
TABLES	
soitemdata = lt_soitemdata.	

```
[...]

ENDMETHOD.
```

In the example of the data provider `$expand` implementation it is possible to fetch application data including Sales Order entity set and associated Sales Order Item entity set using only one function call. In such a use case data provider `$expand` might lead to better performance than framework `$expand`.

Initial Handling for \$expand

When `$expand` is used the backend framework returns the "inline" data to the hub framework in the form of a table containing a flag which shows whether an "inline" structure is initial or not. Thus you can see whether this "inline" data is initial or not.

Related Information

[/IWBEP/IF_MGW_APPL_SRV_RUNTIME](#)
[/IWBEP/IF_MGW_ODATA_EXPAND](#)
[/IWBEP/CL_MGW_ABS_DATA](#)

1.1.4.8 Deep Insert

In some scenarios only updates of hierarchical data are allowed. For example, a Sales Order and a Sales Order Item can only be created together and at the same time, not independently on their own. For this OData Channel provides a **deep insert** functionality: a basic deep insert feature is offered to provide the possibility to the application to create single entities deeply. In addition, deep entities can be created in one activity.

When using deep insert the data need to be nested, that is, a deep structure is expected.

Note

Deep Insert support is only guaranteed for scenarios which are handled fully by the application.

For deep insert interface `/IWBEP/IF_MGW_APPL_SRV_RUNTIME` provides method `CREATE_DEEP_ENTITY`.

The process is as follows:

1. The SAP Gateway framework extracts the expand expression out of the payload (inlined data).
2. Data is sent to the backend.
3. Method `CREATE_DEEP_ENTITY` is called.
4. The application uses `/IWBEP/IF_MGW_ODATA_EXPAND` to validate whether the given expand can be handled.

Initial Handling for Deep Insert

When deep insert is used the backend framework returns the "inline" data to the hub framework in the form of a table containing a flag which shows whether an "inline" structure is initial or not. Thus you can see whether this "inline" data is initial or not.

More Information

For more information see `/IWBEP/IF_MGW_APPL_SRV_RUNTIME` and `/IWBEP/IF_MGW_ODATA_EXPAND`.

1.1.4.9 Error Response Control for Backend Data Provider

Overview

The response of an OData request causing an error looks like this example:

Code Syntax

```
<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <code>/IWBEP/CM_TEA/004</code>
  <message xml:lang="en">'TEAM_012345678' is not in the defined range.</message>
  <innererror>
    <transactionid>4FE1D144B9AD22F2E10000000A420C41</transactionid>
    <errordetails>
      <errordetail>
        <code>/IWBEP/CM_TEA/002</code>
        <message>'TEAM_012345678' is not a valid ID.</message>
        <propertyref>Team/Team_Identifier</propertyref>
        <severity>error</severity>
      </errordetail>
      <errordetail>
        <code>/IWBEP/CM_TEA/004</code>
        <message>Team ID 'TEAM_012345678' is not in the defined range.</message>
        <propertyref>Team/Team_Identifier</propertyref>
        <severity>error</severity>
      </errordetail>
      <errordetail>
        <code>/IWBEP/CX_TEA_BUSINESS</code>
        <message>'TEAM_012345678' is not a valid ID.</message>
        <propertyref/>
        <severity>error</severity>
      </errordetail>
    </errordetails>
  </innererror>
</error>
```

This error response contains the following artifacts:

- Error code
The T100 message class and number of the error message.
In this example it is `/IWBEP/CM_TEA/004`.
- Error message
The text of the error message.
In this example it is `Team ID 'TEAM_012345678' is not in the defined range.`
- Transaction Id
This ID identifies traces belonging to one transaction on all involved systems.
In this example it is `4FDF3748112322F4E10000000A420C41`.

- One or several error detail entries which each contains:
 - Inner error code
The T100 message class and number of the error message or the name of the exception class that contributed this error. In this example it is /IWBEF/CM_TEA/004.
 - Inner error message
The text of the error message.
In this example it is Team ID 'TEAM_012345678' is not in the defined range.
 - Property reference
The XPATH expression (entity type / property name) pointing to the corresponding property for this message if available. In this example it is Team/Team_Identifier or empty.
 - Severity
Possible severities are info, warning and error. In this example it is error.

A backend data provider can control this error by the exception it raises and optionally also by adding messages to the message container of the raised exception.

To trigger an error a backend data provider application raises a business exception or a technical exception:

- A business exception is class /IWBEF/CX_MGW_BUSI_EXCEPTION or a class inheriting from it.
- A technical exception is class /IWBEF/CX_MGW_TECH_EXCEPTION or a class inheriting from it.

Each exception of the exception chain (provided by the previous exception when raising a business or technical exception) will be mapped to an error detail entry, as shown in the following example.

```
<errordetail>
<code>/IWBEF/CX_TEA_BUSINESS</code>
<message>'TEAM_012345678' is not a valid ID.</message>
<propertyref/>
<severity>error</severity>
<target>ID</target>
</errordetail>
```

One of these messages may be marked as the leading message. If a leading message is added to the message container its code and text will be used as the error code and error message of the error response.

➔ Recommendation

We recommend using target. The target allows to relate a detail message to (a part of) an OData resource, or a related OData resource. This is especially needed for errors occurring during form field validation.

The target is always a relative resource path segment:

- For GET, PATCH, PUT, and DELETE requests to a single entity or complex-type instance the target segment is either of the following:
 - Empty - if the error is related to the addressed resource as a whole, for example, a Sales Order.
 - A property path relative to the addressed resource, that is, if a forward slash and the value of target is appended to the path part of the request URL, the result is an OData request URL identifying the target of the error message.
- For POST requests that create a new entity the target segment is relative to the Location response header that identifies the newly created resource, and otherwise follows the rules for GET, PATCH, PUT, and DELETE requests to a single entity.
- For GET requests to a collection of entities the target segment starts with a key segment that identifies a single entity and is then optionally followed by a forward slash and a property path to address a specific part of the entity.

A backend data provider can also use the message container (interface /IWBEF/IF_MESSAGE_CONTAINER) to add one or several messages to the exception it raises.

Options

A backend data provider has several options of how to control the error response:

- Exception only
The backend data provider triggers an exception.
- Exception with message container but without leading message
The backend data provider triggers an exception where the message container contains one or several messages. Only messages with the attribute IS_LEADING_MESSAGE set to FALSE have been added to the message container.
- Exception with message container with leading message
The backend data provider triggers an exception where the message container contains one or several messages. At least one message with the attribute IS_LEADING_MESSAGE set to TRUE has been added to the message container.

Framework Behavior

Depending on the option chosen by the backend data provider the framework behaves as follows.

Exception Only

The text of the triggered exception is used as "Error Message". The exception is also shown as an error detail. If there is an exception chain (the exception referencing another exception via PREVIOUS (that is referencing another exception... that is) each exception of the exception chain is shown as an error detail. Note that if an exception might contain sensitive or too technical information it should **not** be added via PREVIOUS. The inner error code of an exception is the name of the exception class that contributed this error.

Exception Containing the Message Container but Without Leading Message


The error message and the error details are filled the same way as in the case "Exception only". Additionally, every message of the message container is added as an error detail. The inner error code of the message from the message container is the T100 message class and number of that message. These messages can have severity info, warning or error.

Exception Containing the Message Container With a Leading Message

The leading message of the message container is used as the error message. All exceptions of the exceptions of the exception chain are shown as error

details. Additionally, every message of the message container is added as an error detail as described above. If there are no previous exceptions the exception itself is not shown in the error details.

Example of Error Response

 **Sample Code**


```
<?xml version="1.0" encoding="utf-8"?>
<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <code>SG/107</code>
  <message xml:lang="en">Currency AZN is not defined</message>
  <innererror>
    <transactionid>5454F6F183877EC3E10000000A4C4325</transactionid>
    <timestamp>20141107140000.0000000</timestamp>
    <Error_Resolution>
      <SAP_Transaction>Run transaction /IWFND/ERROR_LOG on SAP Gateway hub system and search for entries with the t
      <SAP_Note>See SAP Note 1797736 for error analysis (https://service.sap.com/sap/support/notes/1797736)</SAP_Nc
      <Additional_SAP_Note>See SAP Note 1868586 (https://service.sap.com/sap/support/notes/1868586). This SAP Note
    </Error_Resolution>
    <errordetails>
      <errordetail>
        <code>SG/107</code>
        <message>Currency AZN is not defined</message>
        <propertyref/>
        <severity>error</severity>
        <target>PricingTerms/CurrencyCode</target>
      </errordetail>
      <errordetail>
        <code>SG/122</code>
        <message>Exchange rate zero</message>
        <propertyref/>
        <severity>error</severity>
        <target/>
      </errordetail>
    </errordetails>
  </innererror>
</error>
```

Additional Information

A backend data provider can set the HTTP status code when raising a business or technical exception.

A backend data provider should log all messages relevant for error analysis but that are not relevant for an end user via the logger (class /IWBEF/CL_COS_LOGGER). This writes these logs into the Application Log and optionally also into the Computing Center Management System (CCMS).

It is possible to overwrite the outer message code of the error, for example `<code>Message Code from Exception</code>`:

 **Sample Code**

```
<?xml version="1.0" encoding="utf-8"?>
<error xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
  <code>Message Code from Exception</code>
  <message xml:lang="en">This is a message text of a business exception raised by the provider.</message>
  <longtext_url>/sap/opu/odata/iwbep/message_text;o=G1Y_400_BEP/T100_longtexts(MSGID='%2FIWBEP%2FCM_TEA',MSGNC
  <innererror>
    ...
  </innererror>
</error>
```

The base exception on the backend side /IWBEF/CX_MGW_BASE_EXCEPTION has a new parameter `msg_code` in the constructor, to set the message code. It is also possible to set the message code in the message container: /IWBEF/IF_MESSAGE_CONTAINER~SET_MESSAGE_CODE().

1.1.4.10 \$filter System Query Option APIs

The OData \$filter system query option filters a subset of the entries from the collection that are addressed in the request URL. SAP Gateway provides filter APIs which can be used to query in the SAP backend. Filter expression tree is a tree representation of OData filter system query option provided in the request URI. A visitor interface /IWBEF/IF_MGW_EXPR_VISITOR has been provided to process the filter expression tree. You have the option to:

- Implement the visitor interface /IWBEF/IF_MGW_EXPR_VISITOR in your own way to process the filter expression tree.

- Use the `/IWBEP/IF_MGW_REQ_ENTITYSET` API which is an implementation of the visitor interface `/IWBEP/IF_MGW_EXPR_VISITOR` to get the ABAP Open SQL WHERE clause.

Note

While implementing a query for an entity, you can do a query on the database using Open SQL SELECT syntax if you have direct access to persistency.

Implementing the Visitor Interface `/IWBEP/IF_MGW_EXPR_VISITOR`

Use the method `GET_FILTER_EXPRESSION_TREE` of the interface `/IWBEP/IF_MGW_REQ_ENTITYSET` to get the filter expression tree and then implement the visitor interface `/IWBEP/IF_MGW_EXPR_VISITOR` to process the filter expression tree as per your requirement. You should implement the visitor interface using Visitor design pattern.

To get the filter expression tree, invoke the `/IWBEP/IF_MGW_EXPR_NODE` API in the `GET_ENTITYSET` method of the Data Provider class (DPC).

```
01.Data: lo_filter_tree TYPE REF TO /iwbe/ifu_mgw_expr_node.
  lo_filter_tree = io_tech_request_context->get_filter_expression_tree( ).
```

Using the Interface `/IWBEP/IF_MGW_REQ_ENTITYSET`

The OData \$filter system query option allows clients to filter the set of resources that are addressed by a request URL. Application developers who need to implement a query for an entity in a service and have direct access to persistency can run a direct query on the database using Open SQL SELECT syntax. SAP Gateway provides a feature to get an Open SQL WHERE clause from the OData request that can easily be used in the SELECT query directly. Use the method `GET_OSQ WHERE_CLAUSE` of the interface `/IWBEP/IF_MGW_REQ_ENTITYSET` to get an Open SQL WHERE clause. The Open SQL WHERE clause can be used to do a query in the backend directly using ABAP SELECT syntax.

To get the Open SQL WHERE clause, use the code below in the `GET_ENTITYSET` method of the DPC of an OData service:

```
01.DATA: lv_osql_where_clause TYPE string.
  *---get Open SQL WHERE Clause from $filter-----*
  lv_osql_where_clause = io_tech_request_context->get_osql_where_clause( ).
```

Only a subset of the possible filter expressions can be converted into a WHERE clause.

\$filter on Navigation Property

The system query option \$filter is supported on navigation properties.

Information about \$filter on navigation is provided via the filter expression tree in the backend system for the affected data provider.

Examples of supported URLs containing navigation steps in the \$filter query option:

- `SalesOrder?$filter=TwinEntity eq 'null'`
- `SalesOrder?$filter=TwinEntity/property eq 'xyz'`
- `SalesOrders?$filter=EditState ne 2 or (EditState eq 2 and TwinEntity eq null)`
- `SalesOrder?$filter=TwinEntity/<next_navigation>/...<last_navigation> eq 'null'`

The navigation or complex properties appearing in the \$filter query option will be provided as follows in the filter expression tree::

- `<property name without '>':`
Name of a simple property or a navigation. In case of navigation the property value can only be 'null' and the operator is '=' or '<'.
- `<property name containing one or more '>':`
Name before ':'. This name can be the name of a complex property or a navigation.
Name after ':'. This name can be the name of a simple property or a complex property or a navigation.

1.1.4.11 Map Message Container to Message Protocol Format

Context

If a request is successful, warnings and success messages can be returned to the consumer in the new HTTP header `SAP-Message`. The format of the messages can be either set by the URL option `format=json` or in case of a modifying request with an empty response by using the request header `Accept: application/json`.

Example of adding a message in the header `SAP-Message`:

```
lo_message_container->add_message(
  iv_msg_type           = /iwbe/cl_cos_logger=>warning
  iv_msg_id             = /iwbe/cx_tea_business=>team_id_out_of_range-msgid
  iv_msg_number         = /iwbe/cx_tea_business=>team_id_out_of_range-msgno
  iv_add_to_response_header = abap_true
).
```



Example

`http://<server>:<port>/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Teams('TEAM_03')?$format=json`

```
<notification xmlns:sap="http://www.sap.com/Protocols/SAPData">
  <code>/IWBEP/CM_TEA/004</code>
  <message>Team ID 'XXX_I' is not in the defined range.Team ID 'XXX_I' is not in the defined range.</message>
  <severity>info</severity>
  <target></target>
  <details>
    <detail>
```

```

<code>/IWBEF/CM_TEA/004</code>
<message>Team ID 'XXX_E' is not in the defined range.</message>
<target>Team_Identifier</target>
<severity>error</severity>

</detail>
<detail>

<code></code>
<message>Message text directly set as warning.</message>
<target>Name</target>
<severity>warning</severity>

</detail>
</details>
</notification>

Response in http-header sap-message JSON:
{
  "code": "/IWBEF/CM_TEA/004",
  "message": "Team ID 'XXX_I' is not in the defined range.",
  "severity": "info",
  "target": "",
  "details": [
    {
      "code": "",
      "message": "Message text directly set as warning.",
      "target": "Name",
      "severity": "warning"
    }
  ]
}

End of the code.

```

Related Information

[/IWBEF/IF_MESSAGE_CONTAINER](#)

1.1.4.12 Annotations

SAP:Annotation "Security Level"

OData requests may contain confidential data in the URL which is visible in the browser. If such data shall not appear in the URL, we recommend to use batch requests instead. You can enforce the generation of batch requests with a new SAP Annotation `sap:use-batch="true"` (metadata document). For this a method is available: `model->set_use_batch(abap_true)`. The metadata document will then contain the SAP Annotation `sap:use-batch="true"` in the entity container.

Example

Sample Code

```
<EntityContainer Name="TEST_SRV_Entities" sap:use-batch="true"/>
```

End of the example.

Support OData Element "annotation path"

Based on vocabulary terms annotations can be created. With this feature it is possible to set an annotation path for the vocabulary annotation simple value.

Example

Setting AnnotationPath (backend):

Sample Code

```

lo_ann_target = vocab_anno_model->create_annotations_target( 'EPMDemo.PurchaseOrderItem' ).
lo_annotation = lo_ann_target->create_annotation( iv_term = 'org.example.display.DisplayName' ).
lo_label_elem = lo_annotation->create_labeled_element( 'CustomerFirstName' ).
lo_simp_value = lo_label_elem->create_simple_value( ).
lo_simp_value->set_annotation_path( 'Name/ThisIsMyPath' ).

```

End of the example.

Example

AnnotationPath in metadata definition (SAP Gateway Client example):



Sample Code

```
<Annotations Target="EPMDemo.PurchaseOrderItem" xmlns="http://docs.oasis-open.org/odata/ns/edm">
<Annotation Term="org.example.display.DisplayName"/>
<LabeledElement Name="CustomerFirstName" AnnotationPath="Name/ThisIsMyPath"/>
</Annotation>
</Annotations>
```

End of the example.

1.1.4.13 Extended Support of Long Texts in the Metadata

SAP Gateway provides different UI texts per property, contained in the annotations `sap:label`, `sap:heading` and `sap:quickinfo`. If the property is bound against an ABAP Dictionary data element, then these UI texts are read (if existing) from the underlying data element.

- `sap:label`
Intended for field labels in UIs. Its default source is the **Medium** (green) field label of the underlying data element.
- `sap:heading`
Usually a shorter text than the label text. This text can be used for column headings. Its source is the **Heading** (orange) of the underlying data element.
- `sap:quickinfo`
Usually a longer text that can be used for tooltips or as a secondary help text. Its source is the **Short Description** (red) of the underlying data element.

Data element	BUDAT	Active
Short Description	Posting Date in the Document	
Attributes	Data Type	Further Characteristics
		Field Label
	Length	Field Label
Short	10	Pstng Date
Medium	15	Posting Date
Long	20	Posting Date
Heading	10	Pstng Date

The annotations `sap:heading` and `sap:quickinfo` are only included if their text differs from the always present `sap:label`. Note that the annotation `sap:label` can be also overwritten in Service Builder by redefining services. For more information, see [Redefining OData Services](#).

Furthermore, the metadata document can include classical F1 documentation for ABAP Dictionary data elements (DE) – if this documentation exists. For this, `<Documentation>` elements are included in the metadata document.

By default the `$metadata` request retrieves only the `sap:label` for the property. To show additional text elements a query option needs to be added to the metadata request:

- `sap-documentation=heading,quickinfo`
Additional display of heading and quickinfo
- `sap-documentation=all`
Additional display of heading, quickinfo and F1 documentation

Prerequisites

The properties has to be bound against an ABAP Dictionary data element and the data element has to contain the texts.

Example



Sample Code

Standard \$metadata

```
<Property Name="PostingDate" Type="Edm.DateTime" Precision="7" sap:label="Posting Date" />
```



Sample Code

Extended \$metadata?sap-documentation=heading,quickinfo

```
<Property Name="PostingDate" Type="Edm.DateTime" Precision="7" sap:label="Posting Date" sap:heading="Pstng Date" sap:quickinfo="Posting Date in the Document" />
```



Sample Code

Extended \$metadata?sap-documentation=all

```
<Property Name="PostingDate" Type="Edm.DateTime" Precision="7" sap:label="Posting Date" sap:heading="Pstng Date" sap:quickinfo="Posting Date in the Document" >
<Documentation>
<Summary>Date which is used when entering the document in Financial Accounting or Controlling.</Summary>
<LongDescription>The fiscal year and the period for which an update of the accounts specified in the document or cost elements is made, are derived from the posting date. When entering documents, the system checks whether the posting date entered is allowed by means of the posting period permitted. The posting date can differ from both the entry date (day of entry into the system) and the document date (day of creation of the original document).</LongDescription>
</Documentation>
</Property>
```

1.1.4.14 Soft State Support for OData Services

The so-called "soft state" mode enables the SAP Gateway runtime to process several requests in one ABAP application server session, similar to stateful

behavior. The only difference is the behavior of the application server after the session times out: Instead of breaking the request processing with a time-out exception the server creates a new session and processes the request as usual. For the consumer the change of the application server sessions is transparent and no data is lost on the client session.

The soft state mode should be used for applications built on legacy functionality in the backend, especially when the functionality includes initial loading/caching of big amounts of data for a single request. By using soft state, the resources/functionality which has been loaded during the initial load can be reused for the subsequent requests of the service.

Thus, the main benefit of soft state is a considerable performance optimization of an OData service.

Prerequisites

Data Provider Class (DPC)

In the data provider implementation class of your service make sure that the methods `OPERATION_START` and `OPERATION_END` of the interface `/IWBEP/IF_MGW_SOST_SRV_RUNTIME` are implemented. In case either of them is not redefined and the soft state mode is activated for the service the request processing will be stopped with an exception by the SAP Gateway framework.

- Method `OPERATION_START` checks for data which might still be available in the application buffers, alternatively to load the corresponding resources, that is, fill the cache with data needed for processing.
- Method `OPERATION_END` is supposed to ensure/clear all resources which might get lost if the current session gets closed.

Note

To prepare OData services for working in soft state mode already in SAP Gateway 2.0 SP08 SAP Note [1986535](#) provides interface `/IWBEP/IF_MGW_SOST_SRV_RUNTIME` which is required by the SAP Gateway framework.

However, this SAP Note does not contain a pre-delivery of the soft state processing mode for SAP Gateway 2.0 SP08.

Metadata Provider Class (MPC)

In the metadata provider implementation class of your service make sure that the model has been enabled for soft state mode by using the API `/IWBEP/IF_MGW_ODATA_MODEL~SET_SOFT_STATE_ENABLED`.

Implementation Considerations

When working in soft state processing mode, bear in mind that transactional behavior must not be implemented by using the soft state processing mode and is not provided by the SAP Gateway framework.

- Regardless of soft state mode, SAP Gateway is stateless and also behaves like in a stateless manner. It means that data is not buffered or made available across application server sessions on SAP Gateway. Buffering of data and access to this data is subject to application coding located in the application backend.
- SAP Gateway applications (data provider classes) have to make sure that request processing is not broken due to errors happening in the context of buffering or accessing buffered data in the backend. Example: the data provider class is trying to access data which has been buffered by a previous request in the backend application. In case the buffered data is no longer available as the previous session has expired, the data provider must not throw an exception in this case but retrieve the data again in the same way it would be done in a stateless mode.
- Resources on the server stay occupied as long as the application server session lives. It means that the more services run in soft state mode the more server resources are constantly unavailable. The price you pay for better performance is a higher memory consumption, hence:
 - The soft state timeout should be short.
 - Soft state should only be used in special situations, that is, only for certain entity sets.
- The application server session which is used to run the service in soft state mode is always user-specific.
- Soft state mode can only be activated for a single service, not for a group of services, that is, an application server session can only be used by one OData service.
- The data cached by the data provider of the OData service in the backend during an application server session might get lost or might get outdated.

Constraints

Currently, URLs containing segment parameters such as `;mo, ;o, or ;v=...` cannot be processed in the soft state processing mode, as the path attribute of the context-id sent to the client must not contain a `;`. These kind of URLs are always processed in the stateless mode by the SAP Gateway framework.

The SAP NetWeaver 7.0 component `SAP_BASIS` does not offer security sessions which are prerequisites for soft state. Hence, soft state is not available on SAP NetWeaver 7.0 and/or 7.01. Security sessions are available from SAP NetWeaver 7.0 EHP2 onwards (see SAP Note [1322944](#)). There was also a downport to SAP NetWeaver 7.0 EHP1 (see SAP Note [1477428](#)). However, on SAP NetWeaver 7.0 EHP1 only SAP Gateway 2.0 codeline for SAP NetWeaver 7.0 can be used.

The Internet Communication Manager (ICM) handles soft state requests as if they were stateful requests. The SAP Gateway framework only overwrites the behavior in case of a timeout. See also **Stateless/Stateful Communication** in the SAP NetWeaver documentation. The framework uses an RFC call to communicate with the backend. Nothing special is being done for soft state. Whether work processes of database connections can be maintained or not is therefore not controlled by the framework.

Note that the default setting for session timeout for a stateful connection is set as `00:00:00` in transaction `SICF` in section **Service Data** for an ICF node. This means the session timeout is inactive. The session is terminated after profile parameter `rdisp/plugin_auto_logout` has run. The default setting for `rdisp/plugin_auto_logout` is 30 minutes. If you want to change the time when the session context is terminated, set this profile parameter to another value via transaction `RZ11`. For more information about parameter `rdisp/plugin_auto_logout`, see the parameter documentation in transaction `RZ11`.


Activating Soft State Processing Mode on the Hub System

1. In transaction `SICF` set the session timeout of your service to a value greater than `00:00:00` on the corresponding ICF node.
2. Activate the service for soft state mode in transaction `/IWFND/MAINT_SERVICE` by using the option **Soft State**. The service is active for soft state if the status is set to **Active**.

Possible status:

- **Empty**: in this case the metadata for the service has not been updated on the hub system.

- **Not supported**: in this case the DPC and the MPC do not have the implementations as described in section **Prerequisites**.
 - **Inactive**: in this case the DPC and the MPC do have the implementations as described in section **Prerequisites**, but soft state mode has not been activated for this service in transaction `/IWFND/MAINT_SERVICE`. In this case the soft state based query result cache (SQRC) is also not active.
3. Save your settings.

In addition to activating soft state for individual services, you can also use a central switch which is available in the implementation guide (IMG): In transaction `SPRO` open the `SAP Reference IMG` and navigate to `► SAP NetWeaver ► SAP Gateway ► OData Channel ► Administration ► General Settings ► Enable or Disable Soft State`. The default setting is an enabled soft state for all services. If you disable this centrally with the IMG activity, then those services which are active remain enabled. The switch is only relevant for the runtime. See also [1986626](#) .

Related Information

[Soft State Based Query Result Cache](#)

1.1.4.14.1 Soft State Based Query Result Cache

The soft state based query result cache (SQRC) on the hub caches - if requested by the provider application - the result of a `READ_ENTITYSET` request and applies paging on the result list. Consecutive paging requests for the same query result will then be served from that cache.

A `GET` for an entity set is requested by a client. This will be handled by the SQRC if the following conditions apply:

- The corresponding service has been enabled for soft state in transaction `/IWFND/MAINT_SERVICE` by the provider application supported.

Note

Note that component `IW_FND` and `IW_BEP` both have to be at least on SAP Gateway 2.0 SP09 or component `SAP_GWFND` is on SAP NetWeaver 7.40 SP08.

- SQRC is supported by the provider application.
 - SQRC is not disabled on the backend.
 - Consecutive client requests are a `GET` on the same Entity Set.
- The only difference between the request URLs are the values of the `$top` and `$skip` system query options. The rest of the URL must be identical.

Usage Considerations

It is recommended to use SQRC only if all the following conditions apply:

- The data retrieval is very expensive (slow).
 - The data retrieval cannot be made any faster.
- For example, the data is retrieved from an underlying business framework which is not owned by the provider application. But even in this case it is best to address possible performance issues after they have occurred.
- Avoid caching large Entity Sets.
- If an Entity Set might contain thousands or even hundreds of thousands of entries, filtering must always be used first. The provider application might, for example, refuse the first `READ_ENTITYSET` if no adequate filter has been provided.

Note

As the SQRC is only working if soft state has been enabled, provider applications must also always implement paging themselves and trigger this in case soft state is disabled. With this feature the data returned to the client can always be outdated. There is no cache control mechanism available. Even though using soft state might make the application faster more memory is used.

Disabling SQRC

You can switch off SQRC on the SAP Gateway server. If you experience memory peaks due to the SQRC on the hub server, we recommend to deactivate the SQRC to improve memory consumption. To disable SQRC proceed as follows:

1. In the Implementation Guide (IMG) for SAP NetWeaver navigate to `► SAP Gateway Service Enablement ► Backend OData Channel ► Service Development for Backend OData Channel ► Maintain Service` and click on the `Activity` icon.
2. Enter the technical service name of your service and choose `Display`.
3. Choose `Configuration`.
4. On screen `Change Configuration` activate option `Deactivate Soft State based Query Result Cache`.
5. Save your settings.

The default configuration is an **active** SQRC.

Note

Note that any soft state behavior has to be implemented in the data provider.

Related Information

[Soft State Support for OData Services](#)

1.1.4.15 Search and Open Search Capabilities

On application UIs there is often a search box where users can enter a free text search string. How this kind of search string is used to find adequate results has to be handled by the application. The custom query option search is intended exactly for passing such a free-text search term to the backend and let the backend decide against which properties of each entity in the entity set the term is matched, and in what way. It may also be matched against properties of related entities.



Sample Code

Example of finding all orders with items that refer to a blue product:

```
GET ~/Orders?search=blue
```

End of the example.

SAP Gateway offers OpenSearch capabilities for your service implementation. The search term is in parameter `IV_SEARCH_STRING` of method `GET_ENTITYSET` of interface `/IWBEP/IF_MGW_APPL_SRV_RUNTIME`.

The annotation `sap:searchable="true"` can be added to an entity set. With this annotation the framework tells the client that it can use the custom query option search.



Note

Note that search works just like `$filter`: it will return a subset of the entities that are returned when no search term is specified. And it combines with `$filter` in that only entities that fulfill both conditions are returned.

OpenSearch can be used to tell an OpenSearch client how it can construct URLs with search.

Constraints

If an entity set in Service Builder has been create based on a search help there is at present no code generated that supports search. This has to be done manually by the developer.

Examples



Sample Code

OpenSearch description sample:

```
/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/TravelagencyCollection/OpenSearchDescription.xml
```



Sample Code

Sample search request:

```
/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/TravelagencyCollection?search=Germany
```

1.1.4.16 Simple Field Extensibility

At runtime SAP Gateway classes (model provider class and data provider class) consider the fields in the extension includes in ABAP Dictionary structures. That means that entity sets that are technically based on ABAP Dictionary structures for which SAP provides the option to extend them via append structures will automatically show new properties if the underlying ABAP Dictionary structure has been enhanced by the customer. This process is called Simple Field Extensibility (SFE). Properties that have been created by using SFE contain the annotation `sap:is-extension-field="true"`.

Sample Coding: In this example the type and maxlength of property `AMOUNT11` within entity type `Auto_Expand` is changed.

```
METHOD /iwbep/if_mgw_bd_modify_model~modify_auto_expand_properties.
```

```
DATA: ls_property LIKE LINE OF it_properties,
      lo_property TYPE REF TO /iwbep/if_mgw_odata_property.

IF iv_include_name EQ '/IWBEP/S_TEA_AUTO_EXPAND_I1' AND iv_entity_type_name EQ 'Auto_Expand'.
  LOOP AT it_properties INTO ls_property.
    IF ls_property-internal_name = 'AMOUNT11'.
      ls_property-property->set_type_edm_string( ).
      ls_property-property->set_maxlength( 10 ).
    ENDIF.
  ENDLOOP.
ENDIF.

ENDMETHOD.
```

In the HTTP output, first search for the correct entity type: `Auto_Expand`. When this has been found, scroll down until you find the property modified by the BADI: `'AMOUNT11'`. Verify that the property has the modifications caused by the BADI as shown below.

```
<Property Name="AMOUNT11" Type="Edm.String" MaxLength="10" sap:unit="CURRENCY11" sap:is-extension-field="true"/>
```

1.1.4.17 Conversion Support for Complex Filters

In the past SAP Gateway only offered the conversion of `$filter` expressions into range tables. For certain applications this was not sufficient. Developers had to perform the conversion of complex filter expressions themselves.

Now SAP Gateway supports conversion for complex filter expressions. With this, application developers can now get OpenSQL statements from filter select options.



Sample Code

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=price_1 gt 1600 and currency_1 eq 'JPY' ( ( PRICE_1 > '1600' ) AND ( CURRENCY_1 = 'JPY' ) )
```

More examples of complex filter expressions that are supported:

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=price_1 eq 1600 and currency_1 eq 'JPY' or price_1 eq 1600 and currency_1 eq 'USD'
```

```
<d:unit_2>2M</d:unit_2>
<d:oSQL_Where_Clause>{ ( ( PRICE_1 = '1600.00' ) AND ( CURRENCY_1 = 'JPY' ) ) OR ( ( PRICE_1 = '1600.00' ) AND ( CURRENCY_1 = 'USD' ) ) }</d:oSQL_Where_Clause>
<d:Is_Boolean>true</d:Is_Boolean>
```

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=price_1 gt 1600 and currency_1 eq 'JPY'
```

```
<d:oSQL_Where_Clause>( PRICE_1 > '16.00' ) AND ( CURRENCY_1 = 'JPY' )</d:oSQL_Where_Clause>
<d:Is_Boolean>true</d:Is_Boolean>
<d:TZNTSTMPML>2009-07-01T09:50:00.1234567</d:TZNTSTMPML>
```

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=DATUM eq datetime'2009-07-15T00:00:00'
```

```
<d:oSQL_Where_Clause>( DATUM = '20090715' )</d:oSQL_Where_Clause>
<d:Is_Boolean>true</d:Is_Boolean>
<d:TZNTSTMPML>2009-07-01T09:50:00.1234567</d:TZNTSTMPML>
<d:TZNTSTMPMLL>000000200907010950001234567</d:TZNTSTMPMLL>
<d:TZNTSTMPMS>2009-07-01T09:50:00</d:TZNTSTMPMS>
<d:TZNTSTMPMSL>0001072009095059</d:TZNTSTMPMSL>
<d:TZNTIMELOC>165959</d:TZNTIMELOC>
<d:DATUM>2009-07-15T00:00:00</d:DATUM>
```

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=unit_3 eq 'D10'
```

```
<d:oSQL_Where_Clause>( UNIT_3 = 'D10' )</d:oSQL_Where_Clause>
<d:Is_Boolean>true</d:Is_Boolean>
```

```
/sap/opu/odata/IWBEP/TEA_TEST_APPLICATION/Conversions?$filter=startswith(Id, '1')
```

```
<d:Id>1</d:Id>
<d:price_1>170.59</d:price_1>
<d:currency_1>EUR</d:currency_1>
<d:price_2>1.706</d:price_2>
<d:currency_2>KWD</d:currency_2>
<d:amount_1>1.1235</d:amount_1>
<d:unit_1>DAY</d:unit_1>
<d:amount_2>1.123</d:amount_2>
<d:unit_2>2M</d:unit_2>
<d:oSQL_Where_Clause>ID LIKE '%</d:oSQL_Where_Clause>
<d:Is_Boolean>true</d:Is_Boolean>
```

1.1.4.18 Excel Support

The SAP Gateway framework provides Excel support, that is, support of XLSX.

For OData requests reading an entity set an additional format is supported by SAP Gateway:

- Provided via URL parameter `$format=xlsx`
- Or provided via HTTP header `accept = application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`

In this case the response is not a JSON or an Atom/XML but the binary representation of an Excel file of type `*.xlsx`.

Features

For entity sets the client can request the format XLSX and then retrieves an XLSX instead of a "normal" OData response.

For column headings the property labels are used. If they are initial property names are used.

Complex types are supported. The corresponding properties are shown like simple properties, that is, the table is flattened.

Entity sets for entity types that have a base type are supported.

`$select` is supported. Only columns as requested via `$select` are added. Note that `$select` does not support properties of complex types. But it does support complex properties "on the first level".

The order of `$select` for "not inherited" simple properties is used as the order of the columns.

- Columns for properties of the base type of an entity will be shown in the end (in front of complex types).
- Columns for properties of complex types are always shown in the end.

The headings for properties of complex types are prefixed with the label/name of the corresponding complex property.

The entity container in the `$metadata` document of all services has a new annotation listing the supported formats:

```
sap:supported-formats="atom json xlsx"
```

The time for the XLSX generation is shown in the SAP Gateway performance trace.

The SAP Gateway Client offers `$format=xlsx` via the **Add URI Option** button. For XLSX the SAP Gateway Client shows in the response area the text "Click "Response in Browser" to display data in the right format."

Common HTTP headers (such as `sap-metadata-last-modified`) are returned together with the XLSX file.

Currency and quantity references are provided to the ABAP List Viewer export tool and used there to show the correct number of decimal points.

Properties with the SAP-semantic "URL" are created as "Link to URL" columns in the ABAP List Viewer export tool. The corresponding Excel cells will then contain hyperlinks.

Warnings returned by the ABAP List Viewer export tool are written into the application log. Errors returned by the ABAP List Viewer export tool are returned as HTTP 500 errors.

The HTTP response header `content-disposition` is filled to suggest as file name the entity set name to the client:

```
content-disposition: attachment; filename=Employees.xlsx
```

Restrictions

Note the following restrictions when using XLSX:

- XLSX is only supported for entity sets.
- XLSX is not supported for batch requests.
- XLSX is not support for \$expand.
- XLSX is only supported if the SAP Gateway server is on a SAP NetWeaver 7.40 release.
- There will be no special handling of next links. For example, the framework will not try to create a "larger" Excel by calling the provider application again if the first response contained a NEXT link.
- For column headings the property labels are used. If a label is initial the property's name is used. The heading annotations are not used.
- \$select does not support properties of complex types.
- The order of the properties in the \$select (controlling the column order) is only considered for "not inherited" simple properties.
 - It is not considered for properties of complex types.
 - It is not considered for properties inherited from a base type.
- The data formatting in the XLSX file is not identical to the formatting in the OData XML or OData JSON response.
 - It does not reflect the EDM type definition but is only based on an internal SAP Gateway ABAP representation of the data..
 - Examples:
 - GUID: '005056BA3A451ED48D9E3C2C2FDEB559' vs. '005056BA-3A45-1ED4-8D9E-40ED5ADC3559'
 - Time: '361' vs. '6:01'
 - Left aligned numbers vs. right aligned numbers. Currency and unit amount fields are, for example, left-aligned and not marked as numbers in the XLSX file if the provider application leaves the conversion to the framework.
 - NUMC fields are shown as text in the Excel

1.1.4.19 User Self Service

This section provides information on the new feature from SAP Gateway, User Self Service.

User Self Service is an SAP Gateway feature (similar to the User Management capability of Business-to-consumer (B2C) scenario) using which, SAP Business Suite B2C customers can create users and manage their user profiles using the OData services provided by SAP Gateway. By using User Self Service, your customers can:

- Create a user in the SAP Business Suite system. See [Creating a User Account](#).
- Manage their user profiles.
- Reset the password. [Resetting the Password](#).

OData Services

SAP Gateway provides two OData services to enable User Self Service:

OData Service	Service Document URL
USERREQUESTMANAGEMENT This OData service is always executed in a service user context and is mainly used for creating requests like: <ul style="list-style-type: none">• User creation• Password reset• User creation activation	<a href="http://<hostname>:<port>/sap/opu/odata/IWBEP/USERREQUESTMANAGEMENT">http://<hostname>:<port>/sap/opu/odata/IWBEP/USERREQUESTMANAGEMENT
USERMANAGEMENT This OData service is always executed in the logged in users context and is used for managing the users created in the system like: <ul style="list-style-type: none">• Viewing the user account details• Updating the user account details• Changing the password• Fetching the current user name	<a href="http://<hostname>:<port>/sap/opu/odata/IWBEP/USERMANAGEMENT">http://<hostname>:<port>/sap/opu/odata/IWBEP/USERMANAGEMENT

Related Information

[Configuration Settings for User Self Service](#)

[Security Aspects of User Self Service](#)

1.1.4.19.1 Configuration Settings for User Self Service

Configuring SAP Business Suite system and SAP NetWeaver system for User Self Service.

You must configure the SAP Business Suite system and the SAP NetWeaver system as per the information provided in the following sections:

- [Configuring SAP Business Suite System for User Self Service](#)
- [Configuring SAP NetWeaver System for User Self Service](#)
- [User Self Service IMG Activities](#)

1.1.4.19.1.1 Configuring SAP Business Suite System for User Self Service

Settings on SAP Business Suite system for User Self Service.

Context

Procedure

1. Creating a Secure Type 3 RFC destination between SAP Business Suite system with IWBEP Add-on and SAP NetWeaver system:
 1. Go to transaction SM59.
 2. Choose **ABAP Connections** > **Create**.
 3. Enter a name in the **RFC Destination** field. For example, IWBEP_UM.
 4. Enter **3** in the **Connection Type** field.
 5. Enter an explanatory text in the **Description 1** field and click **Save**.
 6. On **Technical Settings** tab and **Load Balancing** section, select the relevant option according to your system's settings.
 7. In the **Target Host** field, enter the (message) server name of the SAP system.
 8. In the **System Number** field, enter the SAP NetWeaver system number. For example, 00.
 9. On **Logon & Security** tab, enter the SAP system's client number.
 10. Activate Current User.
 11. For **Trust Relationship** activate **Yes** and save the settings.
2. Go to transaction **SPRO**, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **User Self Service Setup** > **Maintain RFC Destinations for User Replication** > Click activity icon.

Note

The implementation type must be IWBEPUM in case you want to use the default User Self Service solution from SAP Gateway.

3. Create a type 3 RFC destination with the name IWBEP_UM_BGRFC for the inbound queue:
 1. Go to transaction SM59.
 2. Choose **ABAP Connections** > **Create**.
 3. Enter IWBEP_UM_BGRFC in the **RFC Destination** field.
 4. Enter **3** in the **Connection Type** field.
 5. Enter an explanatory text in the **Description 1** field and save the settings.
 6. Select the **Special Options** tab and choose **Classic with bgRFC** option as the **Transfer Protocol**.
4. Register the inbound bgRFC destination IWBEP_UM_BGRFC created. This inbound destination is required for executing the user creation and user maintenance process in asynchronous mode:
 1. Go to transaction SGBRFCCONF in the SAP Business Suite system.
 2. Go to the **Define Inbound Dest.** tab and choose **Create**.
 3. In the **Inb. Dest. Name** field, enter IWBEP_UM_BGRFC.
 4. Save the settings.
 5. Go to the **Scheduler: Destination** tab and choose **Create**.
 6. In the dialog box, select **Inbound** as the destination type.
 7. In the **Destination** field, enter IWBEP_UM_BGRFC and save the settings.
 8. In the **bgRFC Destination** screen, choose **Save** to save your customizing settings.
5. Configure the system to send out emails. This is required to send email notifications containing the Activation URL. For more information, see [SAPconnect \(BC-SRV-COM\)](#).

Related Information

[User Self Service IMG Activities](#)

[Configuring SAP NetWeaver System for User Self Service](#)

Configuring SAP NetWeaver System for User Self Service

Settings on SAP NetWeaver system (where SAP Gateway component is installed) for User Self Service.

Context

Procedure

1. Activate the OData services `/IWBEP/USERREQUESTMANAGEMENT (0001)` and `/IWBEP/USERMANAGEMENT` as per the procedure given in [Activate and Maintain Services](#) section.

Note

If you have extended the OData service for `/IWBEP/USERREQUESTMANAGEMENT`, then activate that service.

2. Maintain the logon data for `/IWBEP/USERREQUESTMANAGEMENT`:

1. Go to transaction `SICF`.
 2. Choose `Service` in the `Hierarchy Type` field and click `Execute`.
 3. Enter `/IWBEP/USERREQUESTMANAGEMENT` or the extended OData service in the `Service Name` field.
 4. Double click `/IWBEP/USERREQUESTMANAGEMENT` or the extended OData service which is displayed under `Virtual Hosts / Services` section.
 5. Click `Change` and choose `Logon Data` tab.
 6. Enter the service user name in the `User` field and a password in the `Password` field. This ensures that the service `/IWBEP/USERREQUESTMANAGEMENT` is always executed in the service user context.
 7. Save the settings.
 8. Go to `Service Data` tab and click `GUI Configuration`.
 9. Enter `~CHECK_CSRF_TOKEN` in the `Parameter Name` field and `0` in the `Value` field. This means you are not required to pass X-CSRF token for HTTP POST and HTTP PUT. However, the HTTP header `X-REQUESTED-WITH=XMLHttpRequest` needs to be provided for HTTP POST and HTTP PUT request.
3. Maintain the logon data for `/IWBEP/USERMANAGEMENT`:
1. Go to transaction `SICF`.
 2. Choose `Service` in the `Hierarchy Type` field and click `Execute`.
 3. Enter `/IWBEP/USERMANAGEMENT` or the name of the extended OData service in the `Service Name` field.
 4. Double click `/IWBEP/USERMANAGEMENT` or the extended OData service which is displayed under `Virtual Hosts / Services` section.
 5. Click `Change` and choose `Logon Data` tab under the `Authentication` section.
 6. Choose the option `Internet User`. This will ensure that your customers can logon only with their internet user name or user alias.

Related Information

[User Self Service IMG Activities](#)

[Configuring SAP Business Suite System for User Self Service](#)

1.1.4.19.1.3 User Self Service IMG Activities

List of IMG activities for User Self Service.

User Self Service specific configuration tasks are included in the SAP Gateway Implementation Guide (IMG) which is available in the system. To access these IMG activities, go to transaction `SPRO`, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **User Self Service Setup**. For all steps/activities in the IMG, there is always consumer-independent documentation available. Select the Display icon to view the documentation before executing each activity.

The following are the User Self Service IMG activities:

- Maintain URL for User Account Activation (Mandatory): You can maintain the activation URL of the application you are using to manage your user accounts by using this activity.
- Maintain Number Range Interval for User Self Service (Mandatory): Use this activity to maintain the number range for generating the users in the SAP system.
- Maintain RFC Destinations for User Replication (Mandatory): This IMG activity enables you to replicate the users from the SAP backend to SAP NetWeaver system.
- Maintain User Category (Mandatory): You can maintain the list of user categories for your application using this activity.
- Verify User Request (Optional): This is an activity for implementing the BAdI `/IWBEP/BD_MGW_URM_VERIFICATION`. This BAdI defines the functionality for verifying the information provided during the user request creation.
- Define Notification Process for User Request Management (Optional): This is an activity for the BAdI `/IWBEP/BD_MGW_URM_NOTIFICATION`. It defines the functionality for sending notifications from User Request Management application. User Request Management application has the provision to deliver notifications via email as the standard. You can enhance the solution by adding your own notification mechanism. By default, the notification content is delivered in a certain standard format but you have the provision to customize the format.
- Implement User Management (Optional): This is an activity for the BAdI `/IWBEP/BD_MGW_UM_USR_MANAGER` to manage the users.
- Define Handler for User Management Notification (Optional): This is an activity for the BAdI `/IWBEP/BD_MGW_UM_NOTIFICATION` to notify about the user creation.
- User Request Cleanup (Optional): You can delete the user request which are in process / open / completed / canceled using this activity.

1.1.4.19.2 Creating a User Account

User Self Service allows SAP Business Suite users to create a user account in the SAP system in two steps:

1. Register the user name
2. Activate the user

Once done, the user is created in the SAP Business Suite system and replicated to SAP NetWeaver system.

By default the users created on the default SAP Business Suite system with IWBEP Add-on are replicated to a single SAP NetWeaver system.

Note

User Self Service does not support [Multiple Origin Composition](#) (MOC). This means you can create users on SAP Business Suite system with IW_BEP Add-on followed by the replication of that user to SAP NetWeaver system. However, you can use the BAdIs `Implement User Management` or `Define Handler for User Management Notification` to replicate users to multiple backend systems. To implement the BAdIs, go to transaction `SPRO`, open the SAP Reference IMG and navigate to: **SAP Gateway Service Enablement Backend OData Channel User Self Service Setup** and choose the relevant BAdI. For more information on the BAdI, click the display icon.

Prerequisites

Make sure the following prerequisites are met:

- The SAP NetWeaver system and the SAP Business Suite system is configured as per the information provided in [Configuration Settings for User Self Service](#) section.
- The users should be created with right authorizations, see [Security Aspects of User Self Service](#).

Creating a User Account

1. Registering the user name:

/IWBEP/USERREQUESTMANAGEMENT OData service is used to register a user name. Execute a POST request on the entity UserRequestCollection.

i Note

UserRequest entity in OData service UserRequestManagement is enhanced with a new property Password. With this you have the option to use the Password property while registering a user name. If you are providing the password while registering the user name, there is no need to provide a password while activating the user and vice versa.

```
<entry xml:base="http://<host>:<port>/sap/opu/odata/IWBEP/USERREQUESTMANAGEMENT/"
xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
<updated>2012-10-08T03:25:45Z</updated>
<category term="USERREQUESTMANAGEMENT.UserRequest" scheme="http://schemas.microsoft.com/ado/2007/08/dataservic
<content type="application/xml">
<m:properties>
<d:UserName>B2CUSER1</d:UserName>
<d:FirstName>UsersFirstName</d:FirstName>
<d:LastName>UsersLastName</d:LastName>
<d:EmailAddress>UsersEmailID@xyz.com</d:EmailAddress>
<d:PhoneNumber>123456789</d:PhoneNumber>
<d:UserCategory>001</d:UserCategory>
<d>Password>something</d>Password>
<m:properties>
</content>
</entry>
```

A notification email with the activation link is sent to the user.

i Note

Use the same UserCategory maintained in the Maintain User Category IMG activity. See [User Self Service IMG Activities](#).

2. Activating the user:

/IWBEP/USERREQUESTMANAGEMENT OData service is used to activate a user. Execute a PUT request on the entity

UserRequestActivationRequestCollection with **RequestID** as the key and leave the password property blank if it is already provided during registering the user name (step 1).

```
<entry xml:base="http://<host>:<port>/sap/opu/odata/IWBEP/USERREQUESTMANAGEMENT/"
xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
<category term="USERREQUESTMANAGEMENT.UserRequestActivationRequest" scheme="http://schemas.microsoft.com/ado/2
<content type="application/xml">
<m:properties>
<d:RequestID>005056A502F61EE2B2BA14382200DD3F</d:RequestID>
<d:ActivationKey>3085E67D1FA645C765CDGG899991CEBC28A9B549</d:ActivationKey>
<m:properties>
</content>
</entry>
```

1.1.4.19.3 Resetting the Password

The users may want to reset their password when the user is locked because the number of unsuccessful login attempts exceeded the maximum allowed or when they forget their password or for any other reason. USERREQUESTMANAGEMENT is the OData service used to execute password reset requests.

When a user requests to reset the password:

1. An email notification is sent to the user with an auto generated password and an activation URL.
2. User should click on link to enable the auto generated password.
3. If successful, the auto generated password is set.
4. The user changes the auto generated password set in the next login (recommended).

This is the default behavior of User Self Service when a request for password reset is triggered. The reset request activation call made with entity **UserRequestActivationRequestCollection** of the OData service **USERREQUESTMANAGEMENT** only needs an activation key in the request body, password is not required. If a password is sent by the user, it is ignored and the auto generated password is set.

Sample request body for password reset request activation.

```
<entry xml:base="http://<host>:<port>/sap/opu/odata/IWBEP/USERREQUESTMANAGEMENT/"
xmlns="http://www.w3.org/2005/Atom"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
<category term="USERREQUESTMANAGEMENT.UserRequestActivationRequest"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<content type="application/xml">
<m:properties>
<d:ActivationKey>19E365184B2189BB754589C2EEA4A51B412AD639</d:ActivationKey>
<m:properties>
</content>
</entry>
```

1.1.4.19.4 User Self Service Enhancement Options

BAIs available for User Self Service.

Context

User Self Service provides the following BAIs to extend the solution. All these BAIs are available as IMG activities, the technical names and the corresponding IMG activity names are provided below:

- /IWBE/BD_MGW_URM_VERIFICATION - Verify User Request
- /IWBE/BD_MGW_URM_NOTIFICATION - Define Notification Process for User Request Management
- /IWBE/BD_MGW_UM_USER_MANAGER - Implement User Management
- /IWBE/BD_MGW_UM_NOTIFICATION - Define Handler for User Management Notification

To implement the BAIs, go to transaction SPRO, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **User Self Service Setup** and choose the relevant IMG activity and click **Execute**. For more information on the BAi, click the display icon.

Example - Extending UserRequest Entity: You have the option to extend the UserRequest entity of the USERREQUESTMANAGEMENT OData service with application specific fields like Business partner ID, Meter ID, and so on.

Procedure

1. Enhance the UserRequest structure: The User Request table is provided with an include structure /IWBE/S_MGW_URM_USR_REQ_EXT which can be enhanced and you can add your own include structures as per your requirement.
2. Extend the UserRequest entity in the model:
 1. Create your own metadata provider class and inherit from the super class /IWBE/CL_MGW_URM_MODEL.
 2. Redefine the method DEFINE ().
 3. Inside the method DEFINE (), first call super->define() and then define your own fields for the entity UserRequest using the metadata APIs. See an example code snippet below.

```
method DEFINE.

DATA: lo_entity TYPE REF TO /iwbe/if_mgw_odata_entity_tpy,
      lo_property TYPE REF TO /iwbe/if_mgw_odata_property.

super-> define( ).

lo_entity = model->get_entity_type( 'UserRequest' ).
lo_entity-> create_property( iv_property_name = '<property_name>' iv_abap_fieldname = '<property_abap_fieldname>' ).
"iv_abap_fieldname should be same as present in the field of the structure /IWBE/S_MGW_URM_USR_REQ_EXT

endmethod.
```

4. Save and activate.
3. **Maintain the Model:** The newly created model provider class has to be maintained using the IMG activity which is available in the system.
 1. Go to transaction SPRO, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **Service Development for Backend OData Channel** > **Maintain Models** and click the activity icon.
 2. Enter a technical model name and model version.
 3. Click **Create**.
 4. Enter the name of the model provider class you created above and a description for the same.
 5. Click **Save**.
4. **Register the Service:** The service has to be registered using the IMG activity which is available in the system.
 1. Go to transaction SPRO, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **Service Development for Backend OData Channel** > **Maintain Service** and click the activity icon.
 2. Enter a technical model name and model version.
 3. Click **Create**.
 4. Enter a description for the service.
 5. Enter /IWBE/CL_MGW_URM_DATA in the Data Provider Class field.
 6. Click **Save**.
 7. Click **Assign Model** to assign the model that was created in the Maintain model step.
5. Verify the extended fields: All the extended fields can be verified by implementing the BAi /IWBE/BD_MGW_URM_VERIFICATION. To implement the BAi, go to transaction SPRO, open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway Service Enablement** > **Backend OData Channel** > **User Self Service Setup** > **Verify User Request** and click the display icon. For more information on the BAi, click the display icon.

Related Information

[User Self Service IMG Activities](#)

1.1.4.20 Integration Scenarios

Use

Integration of SAP Gateway with various SAP solutions like SAP BW, SAP Business Suite and so on, helps SAP backend system content to be exposed as OData services. Information on the following integration topics are covered here:

- [Integration of SAP Gateway and SPI](#)
- [Integration of SAP Gateway and GenL](#)
- [Integration of SAP Gateway and SAP Business Warehouse \(BW\)](#)
- [Integration of SAP Gateway With SAP HANA](#)
- [OData Services Consumption and Integration \(OSCI\)](#)

Integration of SAP Gateway and SPI

Use

Service Provider Infrastructure (SPI) is an application and User Interface (UI) technology independent layer for business data exposure, which is used across the whole of SAP Business Suite to build timeless software. Integration of SAP Gateway with SPI allows SPI Application Building Blocks to be provisioned as OData services.

The following topics are covered in this section:

- [Generating an SAP Gateway Service for SPI](#)
- [Mapping Between SPI and OData Model](#)

Prerequisites

SAP NetWeaver 7.4 release comes with SAP_GWFND software component installed by default.

- Make sure you have SAP_GWFND software component installed on your system.
- You should deploy IW_SPI 100 software component either on the SAP NetWeaver 7.4 system or on the SAP backend system.

Note


If the SAP backend system is on lower version of SAP NetWeaver (than 7.40), you should also deploy the software component IW_BEP 100.

Supported Release


Component	Release
ECC 6.0 EhP5	SAP_BS_FND 702 SP09
ECC 6.0 EhP6	SAP_BS_FND 731 SP03

Generating an SAP Gateway Service for SPI

Use

1. Logon to the SAP NetWeaver 7.4 system or SAP backend system.
2. Go to the transaction `SEGW`.
3. You can either [create a new project](#) or open an existing project.
4. If you are editing an existing project, click Display - Change ().
5. Expand the project and in the context menu of `Data Model`, choose `► Redefine ► SPI Service ►`
6. In the `Wizard Step 1 of 3: Redefine Service` screen, enter the following:

Field	Description
<code>RFC Destination</code>	Press F4 and choose the RFC destination of the SPI system or choose <code>NONE</code> in case the SPI system is the same system where <code>IW_SPI 100</code> and <code>IW_BEP 200 SP04</code> are installed.
<code>Application Building Block ID</code>	Press F4 and choose an appropriate ABB ID for which you want to generate the SAP Gateway service. For example, <code>EAMS_OBJK</code>
<code>Generate Metadata Specific to Application</code>	Select this option to enable the generation of metadata specific to application. If this indicator is set, <code>DEFINE</code> Method of Generated Model Provider Class is redefined and the model is defined in this generated method. Else, the generic metadata provisioning options is used to obtain the metadata.

7. Click `Next`.
8. In the `Wizard Step 2 of 3: Redefine Service` screen, enter descriptions for model name and service name.
9. Click `Next`.
10. In the `Wizard Step 3 of 3: Redefine Service` screen, choose the OData artifacts which should be a part of the service you are generating and click `Finish`.
11. Click `Generate Runtime Objects` ().
12. In the `Model and Service Definition` dialog box, you can either:
 - change the `Technical Service Name` to create a new service and, click `Continue`.
 - select the check box `Overwrite Extended Service` to overwrite the existing service and, click `Continue`.
13. Enter the package details and click `Save`.

Once the service is generated, you need to register and activate the service. See, [Service Maintenance](#) for more information.

Note

Make sure that you have [Created an SAP System Alias](#) before activating the service. While creating a SAP system alias, the **Software Version** must be selected as mentioned below depending on the SPI system you are using:

- If the SPI is an SAP_BS_FND 702 SP09 system, choose **Default**.
- If the SPI is an SAP_BS_FND 731 SP03 system, choose **/IWSP/BSFND_731**.

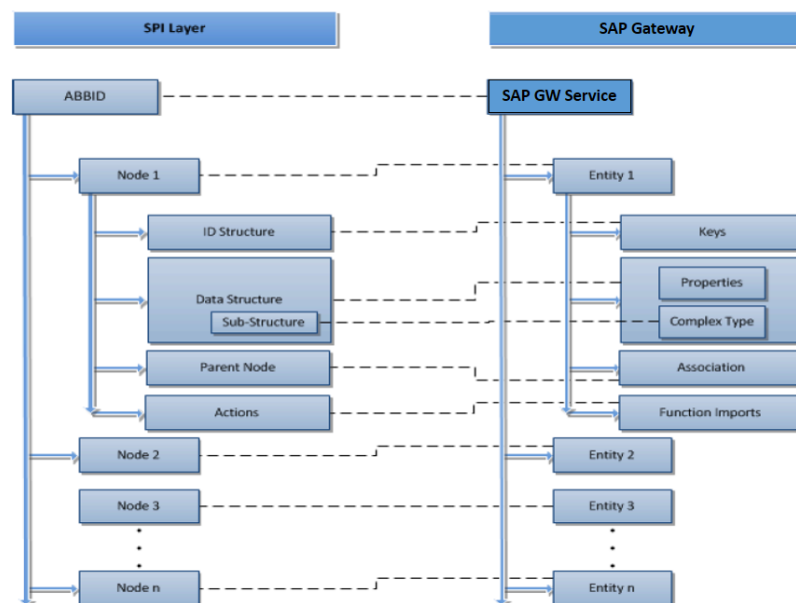
Note

If you have generated a service using SAP_BS_FND 702 SP09 version of the SPI and later you have upgraded your SPI component from SAP_BS_FND 702 SP09 to SAP_BS_FND 731 SP03, your service would work only if the underlying SPI application is not changed. If there is a change in underlying SPI application you must regenerate your service.

1.1.4.20.1.2 Mapping Between SPI and OData Model

Use

The nodes, Data Structure, ID Structure in the SPI Model are transformed to the corresponding entities in an OData Model. Mapping on the same is provided here:



SAP Gateway Relevance Analyzer validates the SPI application internally for SAP Gateway relevance. It runs through the metadata of the SPI model and checks for entities, associations, actions, navigation relevant for SAP Gateway. This runs automatically when you try to create an SAP Gateway Service for SPI. If the SPI application is SAP Gateway relevant, the **SAP Gateway Service Generator for SPI** generates a SAP Gateway service along with the necessary configuration entries. It checks for the following:

Node

1. For each node of SPI layer there is either an ID Structure or an ID Table Type (in SPI layer) which are mapped as Keys in the SAP Gateway layer. The SAP Gateway Relevance Analyzer checks for this and:
 - If either one of them is present, it checks for other conditions
 - If both the ID Structure and the ID Table Type are missing, the node is skipped
2. For each node of SPI layer there is either a Data Structure or a Data Table Type (properties), which is used to get the properties in SAP Gateway layer. The SAP Gateway Relevance Analyzer checks for this and:
 - If either one of them is present, it checks for other conditions
 - If both the Data Structure and the Data Table Type are missing, the node is skipped
3. If the ID Structure or the ID Table type or the Data Structure or the Data Table Type is of Deep type (has a table inside it), then the node is skipped.
4. If the ID Structure or the ID Table type or the Data Structure or the Data Table Type has a Group Name for any of the Named Include then the node is skipped.
5. If the parent node is skipped then all the child nodes are also skipped.

Action

1. If the Action Structure has a Deep Type, the action is skipped.
2. If a node is skipped then all actions of that node are also skipped.

Integration of SAP Gateway and GenIL

Use

Integration of Generic Interaction Layer (GenIL) with SAP Gateway offers a simple and quick way of exposing the business data in the form of OData services.

Check the sections below to know about software requirements, operations supported and constraints for SAP Gateway integration with GenIL, along with this the following topics are covered:

- [Generating an SAP Gateway Service for GenIL](#)
- [Enhancing the Generated Classes](#)
- [Mapping Between GenIL and OData Model](#)

Prerequisites

Make sure that the following system prerequisites are fulfilled:

- You have `SAP_GWFND` installed on your system.
- You should have the following software components deployed either on the SAP NetWeaver 7.4 system or on the SAP backend system:
 - `IW_GIL 100`
 - `WEBCUIF 7.0 SP03` and above

Operations Supported

The table lists all the operations supported for each type of GenIL node:

GenIL Node Type/Operations	Create	Read	Update	Delete	Query
Root Object	✓	✓	✓	✓	✓
Access Object	✗	✓	✗	✗	✓
Dependent Object	✗	✓	✗	✗	✓

✓ Supported

✗ Not supported

Note

The entities corresponding to the root and access objects are marked as `sap:addressable = "true"` and those corresponding to dependent objects are marked as `sap:addressable = "false"`. This means that the query on a dependent object can only be triggered via a navigation property of a root/access object.

Constraints

At present the following constraints apply:

- Currently, the CUD (Create, Update and Delete) operations are not supported on the related objects.
- Filtering is supported only for those properties of an entity (corresponding to the root/access object) which has a similar named field in the attribute structure of the default query.
- Dependent objects which are assigned as related objects to more than one root/access/dependent object in the GenIL model are not supported currently.

Generating an SAP Gateway Service for GenIL

Use

1. Logon to the SAP NetWeaver 7.4 system or SAP backend system where `IW_GIL 100`, and `WEBCUIF 7.0 SP03` are installed.
2. Go to the transaction `SEGW`.
3. You can either [create a new project](#) or open an existing project.
4. If you are editing an existing project, click Display - Change (🔍).
5. Expand the project and in the context menu of `Data Model`, choose **▶ Redefine ▶ GenIL Service ▶**.
6. Choose a GenIL component in the `Wizard Step 1 of 3: Redefine Service` screen and click `Next`.
7. In the `Wizard Step 2 of 3: Redefine Service` screen, enter descriptions for model name and service name.
8. In the `Wizard Step 3 of 3: Redefine Service` screen, choose the OData artifacts which should be a part of the service you are generating and click `Finish`.
9. Choose 'Runtime Artifacts' and click Generate Runtime Objects (🔄).
10. In the `Model and Service Definition` dialog box, you can either:
 - change the `Technical Service Name` to create a new service and, click `Continue`.
 - select the check box `Overwrite Extended Service` to overwrite the existing service and, click `Continue`.
11. Enter the package details and click `Save`.

Once the service is generated, you need to register and activate the service. See, [Service Maintenance](#) for more information.

Related Information

[Enhancing the Classes](#)

1.1.4.20.2.2 Enhancing the Classes

Use

Enhancing the Model Provider Class

The generated Model Provider class for a GenIL based SAP Gateway service fetches the model information at runtime using the GenIL APIs. This class can be enhanced to change the default query for a root/access object in case:

- there are multiple related queries for that object in the corresponding GenIL model

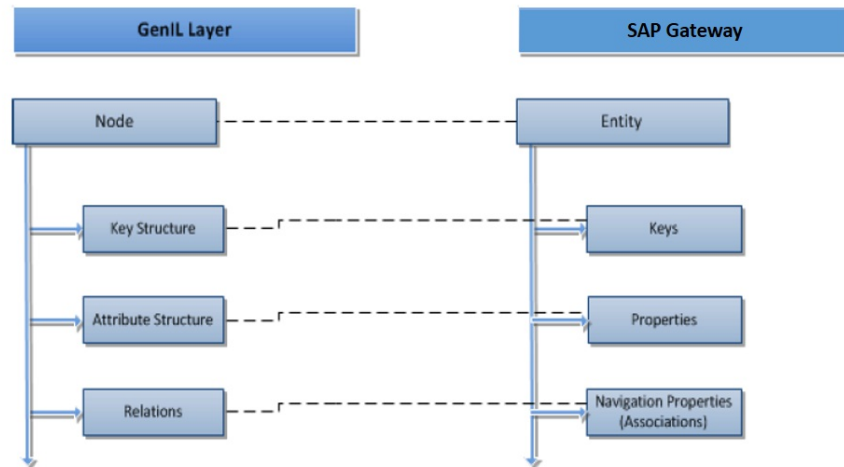
- the default query assigned by the design time logic does not suite the purpose

For this, the method `/IWGIL/IF_GENIL_MODEL_INFO~GET_DEFAULT_QUERY` of the generated model provider class must be modified to return a fixed default query for an entity or to determine a default query at runtime based on the filter options.

1.1.4.20.2.3 Mapping Between GenIL and OData Model

Use

The nodes, relations, queries in the GenIL Model are transformed to the corresponding entities in an OData Model.



Integration of SAP Gateway and SAP Business Warehouse

Use

SAP Gateway and SAP Business Warehouse (BW) Integration enables the SAP BW content to be exposed as OData for SAP Analytic services for light weight mobile consumption. SAP Gateway enables an easy way of integrating SAP BW functionalities by using MultiDimensional eXpressions (MDX) and Easy Query. See, [Generating an SAP Gateway Analytics Service Using MDX and Easy Query](#) section for more information.

Prerequisites

- SAP NetWeaver 7.4 release comes with `SAP_GWFND` software component installed by default.
- You must implement the SAP note [2013195](#) in the SAP backend system.

Note

If the SAP backend system is on lower version of SAP NetWeaver (than 7.40), you should also deploy the software component `IW_BEP_200 SP04`.

Supported Releases

Easy Query

Component	Release
SAP_BW	7.30 SP08 and above 7.31 SP05 and above

MDX

Component	Release
SAP_BW	7.0 and above

Constraints


At present the following constraint apply:

Relevant only for MDX


If the number of records returned from the SAP BW system for any query response is more than one million, SAP Gateway throws an error.

Generating an SAP Gateway Analytics Service Using MDX and Easy Query

Use

1. Logon to the SAP NetWeaver 7.4 system or SAP backend system..
2. Go to transaction `SEGW`.
3. You can either [create a new project](#) or open an existing project.
4. If you are editing an existing project, click Display - Change().
5. Expand the project and in the context menu of `Data Model`, choose `► Redefine ► BW Query Service ►`.
6. In the `Wizard Step 1 of 3: Redefine Service` screen, enter the following:

Field	Description
Access Type	From the dropdown list, choose: <ul style="list-style-type: none">• <code>Controller for MDX (SAP BW)</code> for MDX• <code>Controller for Easy Queries (SAP BW)</code> for Easy Query• <code>Controller for MDX (SAP BW) 2.0</code> - to retrieve the properties defined in the query designer for a query.
RFC Destination	Press F4 and choose the RFC destination of the SAP BW system.
Catalog Name	For MDX - enter a catalog name. For Easy Query - this field is optional.
Query Name	Choose a query name from the F4 help available.

7. Click `Next`.
8. In the `Wizard Step 2 of 3: Redefine Service` screen, enter descriptions for model name and service name.
9. Click `Next`.
10. In the `Wizard Step 3 of 3: Redefine Service` screen, choose the OData artifacts which should be a part of the service you are generating and click `Finish`.
11. Click `Generate Runtime Objects` ().
12. In the `Model and Service Definition` dialog box, you can either:
 - change the `Technical Service Name` to create a new service and, click `Continue`.
 - select the check box `Overwrite Extended Service` to overwrite the existing service and, click `Continue`.
13. Enter the package details and click `Save`.

Once the service is generated, you need to register and activate the service. See, [Service Maintenance](#) for more information.

SAP Gateway With SAP HANA

Use

SAP Gateway with SAP HANA Integration provides a basic integration framework for read-only scenarios based on the approach of OData Channel to expose SAP HANA information models as OData services. SAP HANA Information models are used to create multiple views of transactional data and are the combination of attributes and measures. OData representations of SAP HANA information models can be used for analytical purposes.

Prerequisites

A general knowledge of ABAP OO concepts and tools is required.

For the SAP Gateway with SAP HANA integration the role template `/IWHDB/RT_USER` is available. It contains authorizations object `/IWHDB/HAI`. For more information on roles, see [User, Developer and Administrator Authorizations](#).

Implementation Considerations

For the implementation and usage of SAP Gateway with SAP HANA integration framework you need to maintain the database connection in table `DBCON`.

Furthermore, to declare the maintained SAP HANA connection to the framework you need to implement the enhancement spot `/IWBEP/ES_DESTIN_FINDER`.

Constraints

At present the following integration constraints apply:

- SAP HANA analytic privileges are **not** supported on the ABAP side. Therefore SAP Gateway with HANA can only be used with a service user. Since the ADDBC (ABAP Database Connectivity) interface uses only one defined database user to connect to the SAP HANA database it is not possible to apply an SAP HANA analytical privileges concept within the SAP Gateway integration. As a workaround solution a new authorizations object `/IWHDB/HAI` is available. With the help of this authorization object you can restrict access to the SAP HANA artefacts during runtime. In addition, the model provider class and data provider class contain protected hook methods for authority checks. The default implementation of these authority check methods contain the authority check for the authorization object `/IWHDB/HAI`. Customers have the possibility to enhance these methods by redefinition in the inherited classes. Thus SAP Gateway with HANA offers limited productive usability until the corresponding trust relationship between SAP Gateway and HANA can be supported in SAP NetWeaver Application Server ABAP.
- `$filter`
Filtering for measures is currently **not** supported.

- Input parameters for calculation and analytical views are **not** supported.
- Reference definitions for measure and unit/currency are **not** supported.

Features

SAP Gateway with SAP HANA integration is based on the OData Channel approach for business content enablement. Following this implementation method there are two main components as essential parts of the SAP HANA integration:

- The abstract HDB model provider extracts metadata out the SAP HANA DB repository and adapts it to the OData Channel representation. Metadata retrieval implementation follows a kind of hybrid approach. There is a generic part as well as a specific implementation part.
In order to name specific SAP HANA information models developers needs to re-implement the definition method of the abstract model provider class.
- The generic HDB data provider delegates requests to the SAP HANA DB. Only read scenarios are supported. Runtime data retrieval works generically without a special implementation of sub-classes or configuration. Container structure and metadata data information about which data has to be retrieved will be received with the technical request context information.

With the help of the integration framework the following information models could be exposed as an OData service from the SAP HANA DB:

- **Attribute views**
Attribute views are the reusable dimensions or subject areas used for business analysis.
- **Analytic views**
Analytic views are the multi-dimensional views or OLAP cubes. Using analytic views you analyze values from a single fact table that is based on the related attributes from the data foundation and attribute views.
- **Calculation views**
Calculation views are used to create your own data foundation using database tables, attribute views, analytic views, and calculation views to address a complex business requirement.

Procedure

[Creating and Adding a HANA DB Based SAP Gateway Service](#)

More Information

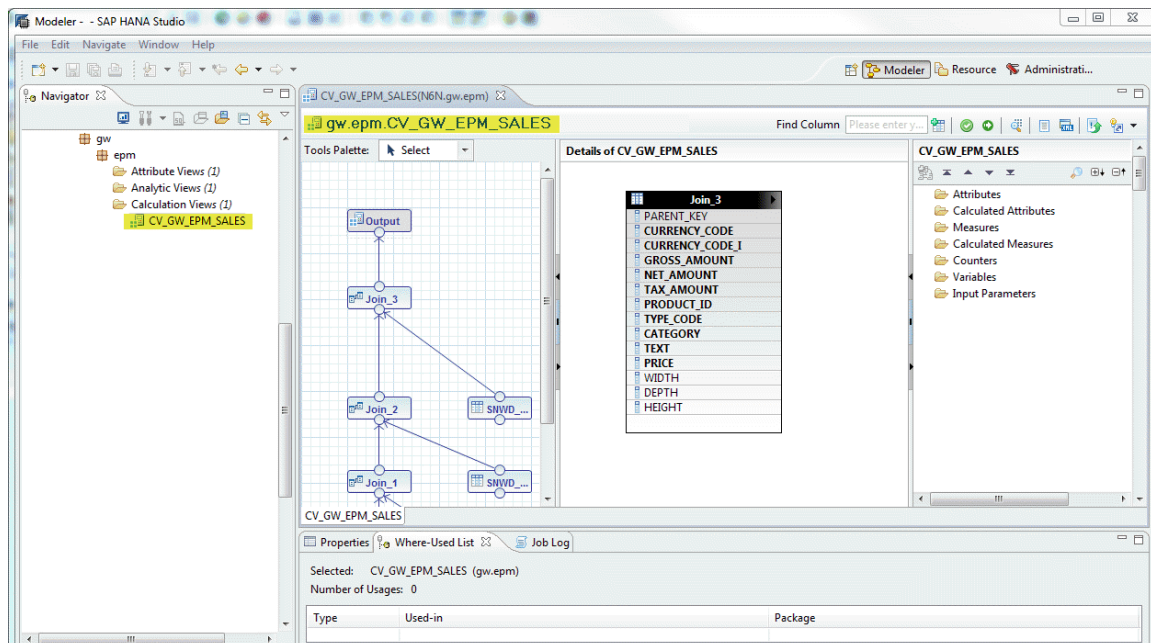
[SAP Gateway with SAP HANA APIs](#)

Creating and Adding a HANA DB Based SAP Gateway Service

Use

The following steps are required to create a SAP Gateway service for SAP HANA information models. The first step contains the integration-specific information, the later steps correspond to those described in the OData Channel quick guide.

1. Define / Choose SAP HANA artifacts.

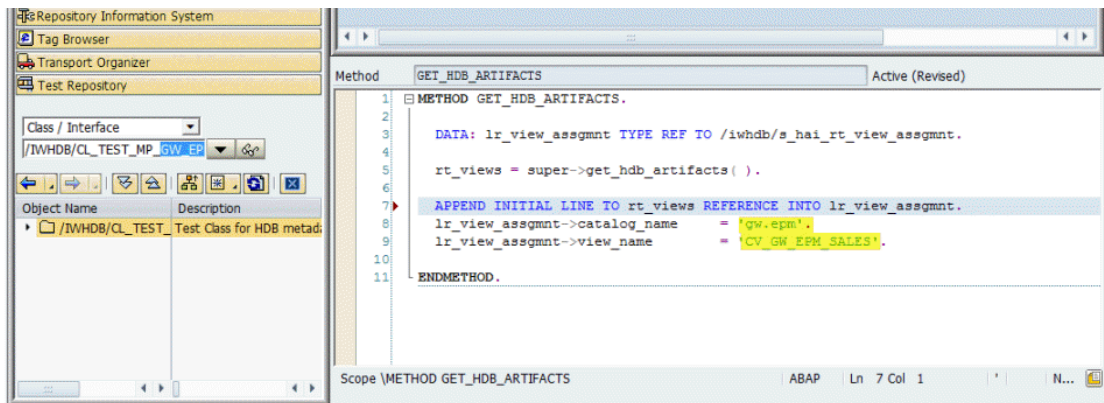


2. Create the model provider class.

In transaction SE24 create a new ABAP OO class based on superclass /IWHDB/CL_HAI_RT_ABS_MODEL as the model provider class (MPC).

1. Redefine and implement method GET_HDB_ARTIFACTS.
2. Set catalog, view and optionally entity set name for the SAP HANA artifacts.

Class Builder: Class /IWHDB/CL_TEST_MP_GW_EPM_SALES Change			
		Pattern	Pretty Printer
		Signature	
		Public Section	Protected Section
		Private Section	
MIME Repository	Ty. Parameter	Type spec.	Description
Repository Browser	VALUE(RT_VIEWS)	TYPE /IWHDB/T_HAI_RT_VIEW_ASSGMNT	Entity Type and View assignment



3. Register your model / service.

1. Define your model.
 1. Create a new model.
 2. Assign the MPC and enter a model description.
 3. Save your model.
2. Define your service.
 1. Create a new service.
 2. Assign the generic DPC /IW_HDB/CL_HAI_RT_DATA and enter a service description.
 3. Save your service.
 4. Assign your model to your service.
 5. Save the service to model assignment.

More Information

For more information on the SAP Gateway with SAP HANA integration, see [SAP Gateway With SAP HANA](#).

For general information on how to create an OData Channel based service, see [Creating OData Channel Content](#).

SAP Gateway with SAP HANA APIs

Use

Both integration classes for metadata and runtime data retrieval are created based on the OData Channel metadata and runtime data APIs. Therefore, only SAP HANA integration specific methods are described below.

The following methods are provided for application development, respectively enhancements:

- Abstract model provider class [/IW_HDB/CL_HAI_RT_ABS_MODEL](#)
- Generic data provider class [/IW_HDB/CL_HAI_RT_DATA](#)

More Information

[OData Channel APIs](#)

1.1.4.20.4.2.1 /IW_HDB/CL_HAI_RT_ABS_MODEL

Use

This is the abstract model provider class (MPC).

Interface Method /IW_HDB/IF_HAI_RT_MED_SRC_INF~GET_HDB_VIEW_ENTITY_ASSIGNMENT

This method returns the list of SAP HANA views with entity type assignments. The method comes from the interface [/IW_HDB/IF_HAI_RT_MED_SRC_INF](#) and is used especially to provide the data source information to the runtime.

Parameter	Type	Description
RT_HDB_ENTITIES	TY_T_HDB_VIEW_ENTITY_ASSIGN	

Interface Method /IW_HDB/IF_HAI_RT_MED_SRC_INF~GET_SKIPTOKEN_RECORDS_NUMBER

This method returns the number of records per page / skip token. The method comes from the interface [/IW_HDB/IF_HAI_RT_MED_SRC_INF](#) and is used especially to provide the server-side paging to the runtime.

Parameter	Type	Description
RV_REC_NUMBER	I	

Protected Method GET_HDB_ARTIFACTS

This method returns the list with SAP HANA artifacts. It is called in the standard / default implementation of the interface method [/IW_HDB/IF_HAI_RT_MED_SRC_INF~GET_HDB_VIEW_ENTITY_ASSIGNMENT](#) in class [/IW_HDB/CL_HAI_RT_ABS_MODEL](#). Therefore the application

developer usually only needs to implement this method to fill the specific SAP HANA artifacts. In case of special assignments of entity names, entity sets and SAP HANA artifacts the method `/IWHDB/IF_HAI_RT_MED_SRC_INF~GET_HDB_VIEW_ENTITY_ASSIGNMENT` needs to be re-implemented.

Parameter	Type	Description
RT_VIEWS	/IWHDB/T_HAI_RT_VIEW_ASSGMNT	

Protected Method CHECK_AUTHORITY

This method checks the authorization to access SAP HANA artifacts. The standard implementation of this method contains a check for the authorization object `/IWHDB/HAI`. An application developer can re-implement this method and define customer-specific authorization checks.

Parameter	Type	Description
IV_CATALOG_NAME	/IWHDB/HANA_CAT_NAME	SAP HANA DB catalog name
IV_VIEW_NAME	/IWHDB/HANA_OBJECT_NAME	SAP HANA DB view name

1.1.4.20.4.2.2 /IWHDB/CL_HAI_RT_DATA

Use

This class is the generic data provider class (DPC).

Protected Method CHECK_AUTHORITY

This method checks a user's authority to access SAP HANA artifacts.

A standard implementation of this method contains a check for the authorization object `/IWHDB/HAI`. The application developer is able re-implement this method and define customer-specific authority checks.

For more information on SAP Gateway roles, see [User, Developer and Administrator Authorizations](#).

Parameter	Type	Description
IV_CATALOG_NAME	/IWHDB/HANA_CAT_NAME	SAP HANA DB catalog name
IV_VIEW_NAME	/IWHDB/HANA_OBJECT_NAME	SAP HANA DB view name

1.1.4.20.5 OData Services Consumption and Integration

Use

Integration of SPI, GenIL, Analytics and HANA with SAP Gateway allows consumption of business data and generates OData services. With OData Services Consumption and Integration (OSCI), SAP Gateway can now consume an external OData service and generate an SAP Gateway compliant OData Service.

The following topics are covered in this section:

- [Creating an RFC Destination](#)
- [Generating an SAP Gateway Service for OData](#)
- [Enabling Media Links](#)

Prerequisites

You have an SAP NetWeaver system where software component `SAP_GWFND` is deployed.

Constraints

You can find a list of current constraints in SAP Note [1574568](#) .

1.1.4.20.5.1 Creating an RFC Destination

Use

You should have an RFC destination configured from the SAP NetWeaver system to the external server (type G) or the ABAP system (type H) before generating the OData service. Follow the procedure given below to configure an RFC destination for either 'G' or 'H' connection type.

1. Logon to the SAP NetWeaver system where `SAP_GWFND` is installed.
2. Go to transaction `SM59`.
3. Choose **Create** and enter the following:

Field	Description
RFC Destination	Enter an RFC destination name. For example, <code>ODATA_RFC</code> .
Connection Type	Choose: <ul style="list-style-type: none">• H for HTTP Connection to ABAP System• G for HTTP Connection to External Server
Description	Enter a description.

4. Go to the **Technical Settings** tab and enter the following:

Field	Description
Target Host	For Type G - Enter the domain name. For example, in the external OData



	service, http://services.odata.org/Northwind/Northwind.svc/ , 'services.odata.org' is the target host. For Type H - Enter the host IP
Path Prefix	Enter any additional directories, if relevant.






5. If relevant, select the **Login & Security** tab and enter the security settings as required.
6. Save the RFC destination.

Continue with [Generating an SAP Gateway Service for OData](#)

Generating an SAP Gateway Service for OData

Use

1. Logon to the SAP NetWeaver 7.4 system.
2. Go to transaction `SEGW`.
3. You can either [create a new project](#) or open an existing project.
4. If you are editing an existing project, click Display - Change ().
5. Expand the project and in the context menu of **Data Model**, choose **► Redefine ► OData Service** .
6. In the **Wizard Step 1 of 3: Redefine External OData Service** screen, enter the following:

Field	Description
HTTP Destination	To select the HTTP Destination: <ol style="list-style-type: none"> 1. Press F4 on the keyboard and choose the connection type. <ul style="list-style-type: none"> • G for HTTP Connection to External Server • H for HTTP Connection to Internal Server 2. Choose Continue. 3. Choose the RFC destination from the list.
Service Namespace	Enter the appropriate Service Namespace. For example, in the external OData Service, http://services.odata.org/Northwind/Northwind.svc/  , 'Northwind' is the Service Namespace <div>  Note You can leave this field blank if the source OData service you are using does not have a Service Namespace. </div>
Service Name	Enter the appropriate Service Name. For example, in the external OData Service, http://services.odata.org/Northwind/Northwind.svc/  , 'Northwind.svc' is the Service Name <div>  Note You can leave this field blank if the source OData service you are using does not have a Service Name. </div>
HTTP Header	Select the check box. Enter the HTTP Header names which will be passed / received by the consuming applications for the external OData Service you are using. <div>  Note You can pass either the request header name or response header name or both depending on the external OData service you are using. </div>

7. Click **Next**.
8. In the **Wizard Step 2 of 3: Redefine Service** screen, enter descriptions for model name and service name.
9. Click **Next**.
10. In the **Wizard Step 3 of 3: Redefine Service** screen, choose the OData artifacts which should be a part of the service you are generating and click **Finish**.
11. Once the service is generated, you need to register and activate the service. See, [Service Maintenance](#) for more information.

1.1.4.20.5.3 Enabling Media Links

Media type is only supported for feed and entry. You should redefine the method `DEFINE()` of the model provider extension class generated by the SB to avail the media link functionality.

Go to the generated model provider extension class and redefine the `DEFINE()` method using the following code:

```
DATA:
    lo_entity_type      TYPE REF TO /iwbep/if_mgw_odata_entity_typ,
    lo_property         TYPE REF TO /iwbep/if_mgw_odata_property.

CALL METHOD super->define( ).
lo_entity_type = model-
>get_entity_type( iv_entity_name = <SOURCE_ENTITY> ).
lo_property = lo_entity_type-
>get_property( iv_property_name = <SOURCE_PROPERTY> ).
lo_property->set_as_content_type( ).
End of the code.
```

Where, `SOURCE_ENTITY` is the entity name of the source service which has a media type (`HasStream = True`) and `SOURCE_PROPERTY` is the property of

SOURCE_ENTITY which has the content type information. For example, in the code given below, SOURCE_ENTITY is 'Image' which has "HasStream = True". The property "MimeType" of entity Image provides the MIMETYPE of the content such as JPEG, GIF, PNG....

```
DATA:
    lo_entity_type      TYPE REF TO /iwbsp/if_mgw_odata_entity_typ,
    lo_property         TYPE REF TO /iwbsp/if_mgw_odata_property.

CALL METHOD super->define( ).

lo_entity_type = model-
>get_entity_type( iv_entity_name = 'Image' ).
lo_property = lo_entity_type-
>get_property( iv_property_name = 'MimeType' ).
lo_property->set_as_content_type( ).
End of the code.
```

1.1.4.20.6 Model Composition for Integration Scenarios

Model Composition allows for mashing up multiple services.

Context

Model composition is provided for complex scenarios for the integration topics, such as SAP Business Warehouse, GenIL and SPI. It allows mashing up multiple services within SAP_GWFND. Thus services and model implementations can be reused for other use cases. You can, for example, compose two or more services to form a completely new service without having to change the existing services.

The resolution of the navigation (to which end point and service the request is routed finally) can be overruled by custom coding. This resolution of the proper navigation is done on the SAP Gateway hub system with only a minimal footprint on the hub.

For model composition an IMG activity is available in the SAP Gateway hub system: In transaction SPRO open the SAP Reference IMG and navigate to ► **SAP NetWeaver** ► **SAP Gateway** ► **OData Channel** ► **Composition** ► **Create BAdI Implementation for Model Composition** ►

Prerequisite

You need to have software components SAP_GWFND installed in your system landscape.

Procedure

1. Define your model composition

Out of two or more existing models/services create one new service with additional semantics and navigation options. For this you use method INCLUDE_MODEL_BY_NAME of interface /IWBEF/IF_MGW_ODATA_MODEL.

Example:

```
model->include_model_by_name(
    iv_service_external_name = 'RMTSAMPLEFLIGHT'
    iv_service_version       = '0001'
    iv_model_tech_name       = '/IWBEF/OM_SAMPLE_EXTD_SFLIGHT'
    iv_model_version         = '0001' ).
```

2. Define the navigation and association between two or more models (this step is optional).

1. Define the association between the current model and the included one by using the following:

Example:

```
iv_def_assoc_set = abap_false
lo_association = model->create_association(
    iv_association_name = 'ConnectionCo2'
    iv_left_type        = 'EXT_ZML_FLIGHTCO2'
    iv_right_type       = 'Flight'
    iv_right_card       = cardinality_entity
    iv_left_card        = cardinality_entity
    iv_def_assoc_set    = abap_false
).
```

2. Define the proper association set:

Example:

```
model->create_association_set(
    iv_association_set_name = 'ConnectionCo2_ASSOC_SET'
    iv_left_entity_set_name = 'EXT_ZML_FLIGHTCO2Collection'
    iv_right_entity_set_name = 'FlightCollection'
    iv_association_name     = 'ConnectionCo2'
).
```

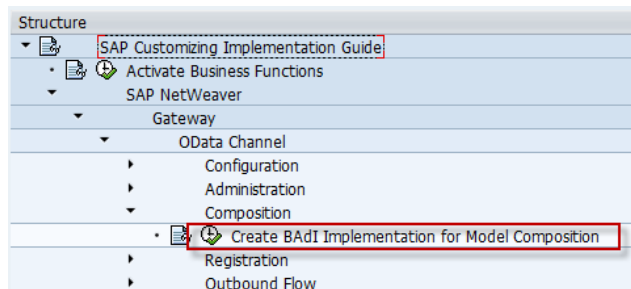
3. Define the navigation property between the two models by using the new association:

Example:

```
lo_entity->create_navigation_property(
    iv_property_name      = 'co2flight'
    iv_association_name   = 'ConnectionCo2'
).
```

3. Implement a custom navigation functionality on the SAP Gateway hub system if the default handling is not sufficient (this step is optional).

In transaction SPRO open the SAP Reference IMG and navigate to ► **SAP NetWeaver** ► **SAP Gateway** ► **OData Channel** ► **Composition** ► **Create BAdI Implementation for Model Composition** ►



1. Carry out the activity by choosing the Activity icon.
If a popup appears with the information that a BAdI implementation is created for BAdI /IWFND/BD_MGW_MDL_COMPOSITION of enhancement spot /IWFND/ES_MGW_MDL_COMPOSITION, choose Continue (Enter).
2. Create the BAdI implementation by entering the Enhancement Implementation and a Short Text and then Creation of Enhancement (Enter).
3. Enter the package name for your implementation. The enhancement implementation for the BAdI displays.
4. Define a filter for your purpose.
5. Implement the navigation using SE80.
All changing parameters can be modified. This means that the navigation path, the table of key properties, as well as the select option table can be changed within the implementation.

Related Information

[Integration of SAP Gateway and SAP Business Warehouse](#)

[Integration of SAP Gateway and GenIL](#)

[Integration of SAP Gateway and SPI](#)

1.1.4.21 Subscription and Notification Flow

Use

In addition to the normal [OData Channel](#) capabilities, SAP Gateway also offers subscription and notification flow functionality.

- [Subscription and notification flow for push oriented scenarios](#)
- [Subscription and notification flow for pull oriented scenarios](#)
- [Subscription Management \(Subscription by administrator on behalf of the users\)](#)

Subscription

Subscription is always done to a business object collection using a filter.



Example

A user may subscribe to all customers belonging to California and thus be notified as soon as a new California customer is created in the backend system or if there are any changes in any California customer.

The most important particular cases of the filter are:

- Subscription to the whole collection (the filter is empty)
- Subscription to a specific business object instance (the filter contains just an ID)

Subscription and Notification Formats

In order to enhance the usability of the Push and Pull Notifications in SAP Gateway, the users now have the option to receive Notifications in two formats:

- ATOM XML
- JSON

Note

Push - While subscribing for a push notification, the value of `Accept` HTTP header must be given as `application/json` to receive notifications in JSON format. Else, by default you receive notifications in ATOM XML format.

Pull - While subscribing for a pull notification, it does not matter what is passed in the `Accept` HTTP header. While pulling the notifications from the SAP Gateway system through the service, `NOTIFICATIONSTORE`, the value of `Accept` HTTP header should be the appropriate value in which the Notifications are desired.

Feed

A feed is simply a collection of notifications for an end-user. It is the result of user's subscriptions to different collections. The notifications may be pushed automatically to the end-user or pulled.



Example

Example of a feed on a mobile device:

- A Travel Request of \$1500 from Martin Lacasse was received
- Sales Order 123 was updated
- The address of Customer 456 has changed
- A new track inspection was assigned to your work center

Subscription Process Flow

1. To subscribe, a consumer sends a subscription request to SAP Gateway specifying a user and subscription content (collection, filter, delivery method, and so on).
2. SAP Gateway has to validate the request and to forward it further to backend.
3. If the subscription is accepted by the backend then it is saved by SAP Gateway in the subscription persistence layer and an acknowledgment is sent to the consumer. The process flow to unsubscribe is similar.

Notification Process Flow

1. The backend tracks the changes in business objects and sends a change notification to SAP Gateway as soon as any change/create/update happens in the business object collection subscribed to.
2. SAP Gateway receives the notification, and forwards it to a consumer using the appropriate format and delivery method. If the consumer is a connectivity server it has its own logic to deliver the notification to end-devices.

Subscription Requests

Create Subscription

The delivery address must have one of the following formats:

- urn:sap-com:channel :<RFC destination>:<request URI>
- http://... or https://...

Delete Subscription

Note that a subscription ID must exist for your user.

```
Example:
DELETE: http://<server name>:<port number>/sap/opu/odata/iwfd/RMTSAMPLEFLIGHT/SubscriptionCollection (value='005056B40E791DEFB0A1

Request Header:
Name = X-Requested-With, value = XMLHttpRequest
```

Get List of Subscriptions

Note that the list contains subscriptions for the current user only.

```
GET: http://<server name>:<port number>/sap/opu/odata/iwfd/RMTSAMPLEFLIGHT/SubscriptionCollection
Request Header:
Name = X-Requested-With, value = XMLHttpRequest
End of the source code.
```

Get Subscription Details

Note that a subscription ID must exist for your user.

```
GET: http://<server name>:<port number>/sap/opu/odata/iwfd/RMTSAMPLEFLIGHT/SubscriptionCollection
(value='005056B40E791DEFB0AD9237BD0657E1',scheme_id='IWF_SUBSCRIPTION',scheme_agency_id='abc_004')
Request Header:
Name = X-Requested-With, value = XMLHttpRequest
```

1.1.4.21.1 Subscription and Notification Flow for Push Oriented Scenarios

Use

In subscription and notification flow for push oriented scenarios:

- Users can subscribe to receive notifications if any changes are made for any subscription-enabled entity type.
- These events are sent by the SAP Backend and published via the SAP Gateway system using HTTP. Users will get notifications whenever any changes are done to the subscribed entity types.

The following topics are covered:

- [Creating a New OData Channel Push Application](#)
- [Enabling Push for an Existing OData Channel Application](#)
- [Notification Content Publisher](#)
- [Notification Content Formatter](#)

1.1.4.21.1.1 Creating a New OData Channel Application for Sending Notifications

Use

To create a new OData Channel push application, you have to carry out different steps in the:

- [SAP Backend System](#)
- [SAP Gateway System](#)

1.1.4.21.1.1.1 Configurations on the SAP Backend System for Push Oriented Scenarios

Use

Prerequisites

Make sure that the following prerequisites are fulfilled:

- The software component `SAP_GWFND` is installed on your system.
- Configured a supervisor destination for the bgRFC to receive the configuration settings for the bgRFC scheduler. For more information, see [Configure the bgRFC Supervisor Destination](#).
- An RFC destination from the SAP backend system to the SAP Gateway system has been created with transaction `SM59`. Provide a service user and a password for the destination.
For more information, see [Maintaining Outbound bgRFC Queue from SAP Backend System to the Hub System](#).
- [Maintaining Inbound bgRFC Queue on the Hub System](#).

Note

This is optional and is required only if a reliable delivery of notifications to the consumer is expected.

- An SAP system alias for the SAP Gateway system has been created in the backend system. For this an IMG activity is available in the system: In transaction `SPRO` open the SAP Reference IMG and navigate to ► [SAP NetWeaver](#) ► [SAP Gateway Service Enablement](#) ► [Backend OData Channel](#) ► [Connection Settings to SAP Gateway](#) ► [SAP Gateway Settings](#) ►.

Defining the Metadata

1. Create a class for metadata definition.
2. Inherit from the SAP Gateway object/meta model super class `/IWBEP/CL_MGW_PUSH_ABS_MODEL`.
3. Redefine method `DEFINE()` in your class to create your own implementation,
4. Add `SUPER->DEFINE()` as first statement to your implementation to define the subscription data for your application.
5. Implement metadata definition.
6. Call method `SET_SUBSCRIBABLE()` for each entity type in your implementation for which you want subscription and notification to be enabled.

Example

Class `/IWBEP/CL_MGW_MED_SFLIGHT` is delivered as an example implementation of a Subscription/Notification metadata definition.

Creating a Data Provider Class

1. Create a class for data provisioning.
2. Inherit from the SAP Gateway data provider super class `/IWBEP/CL_MGW_PUSH_ABS_DATA`.
3. Redefine abstract method `CHECK_SUBSCRIPTION_AUTHORITY` in your class to implement your checks for subscription authorization.
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.
5. Implement the data provider.

Example

Class `/IWBEP/CL_MGW_RT_SFLIGHT` is delivered as an example implementation of a subscription/notification data provider.

Sending Notifications from Backend to Consumer via SAP Gateway

1. Call static method `/IWBEP/CL_MGW_SUB_REGISTRY=>GET_SUBSCRIPTION_REGISTRY()` to get the instance of the subscription registry.
2. Call method `QUERY_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 `/IWBEP/CL_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.

Example

In the system the ABAP report `/IWBEP/R_MGW_PUSH_TEST` is available to demonstrate the notification transmission.

Continue with [Configurations on the SAP Gateway System for Push Oriented Scenarios](#)

Configurations on the SAP Gateway System for Push Oriented Scenarios

Use

1. Setting Up Consumer-Specific Configuration on SAP Gateway: - Logon to your SAP Gateway host and in transaction `SM59` create an RFC destination of type G (HTTP Connection to External Server) pointing to your PC.
2. Subscribing for sample user collection: - Provide a service user and password for the destination(s) To test subscriptions/notifications post an XMLHttpRequest to the SubscriptionCollection of your OData Channel application URL.

Request

```

POST /iwfnd/RMTSAMPLEFLIGHT/SubscriptionCollection HTTP/1.1
Host: <host>:<port>
accept: application/atom+xml
content-type: application/atom+xml
Content-Length: 634
X-Requested-With: XMLHttpRequest

<?xml version="1.0" encoding="utf-8"?>
entry xml:base="http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/" xmlns="http://www.w3.org/2005/Atom"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns:d="http://schemas.microsoft.com/ado/2007/08/d
<id>http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/SubscriptionCollection('00163EA725801ED29598CE35319B1772')</id>
<title type="text">My sub</title>
<updated>2013-01-02T10:23:26Z</updated>
<author>
<name>SAPUSER</name>
</author>
<category term="RMTSAMPLEFLIGHT.Subscription" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<link href="SubscriptionCollection('005056A509B31ED2849EFB180361F2D6') " rel="edit" title="Subscription"/>
<content type="application/xml">
<m:properties>
<d:ID>00163EA725801ED29598CE35319B1772</d:ID>
<d:deliveryAddress>HTTP://10.xx.xxx.000:xxxx<d:deliveryAddress>
<d:persistNotifications>false</d:persistNotifications>
<d:collection>FlightCollection</d:collection>
<d:filter/>
<d:select/>
<d:changeType>created</d:changeType>
</m:properties>
</content>
</entry>

```

The different parts of the source code are explained in the following sections.

- **<d:deliveryAddress>**

The delivery address must have one of the following formats:

- urn:sap-com:channel:<RFC destination>:<request URI>
- http(s)://<host>:<port>/...

In both cases an RFC destination of type G (HTTP destination to external server) must exist on SAP Gateway either with the name <RFC destination> or targeted to the <host>:<port>.

The comparison is case-insensitive and it is performed for strings (that is, no DNS names resolution, and so on).

For example, if you want to subscribe with delivery address `http://endpoint.somewhere.com:8000/application/user-inbox` there must be an RFC destination with target host `endpoint.somewhere.com` and target port `8000`.

- **<d:filter>** - The filter attribute must follow the OData `$filter` notation (see <http://www.odata.org/developers/protocols/uri-conventions#FilterSystemQueryOption> 📌) and may combine any fields using the AND operation only. SAP Gateway does not evaluate any filter criteria.
- **<d:changeType>** - The changeType attribute must be one of the following - "created", "updated" "failed" or "deleted". changeType property allows you to subscribe for notifications based on a particular operation performed in the SAP backend. For example, if you want to subscribe for notifications only when a create operation is done, then the changeType would be "created". Users will have to develop application specific handling for sending notifications based on the subscribed changeType. Hence, for the above example given, the user is not notified when an update or a delete operation is performed on the object for the subscribed collection in the SAP backend.

Testing Notifications

Before you can test your notifications make sure that you have already created a subscription in the first place.

1. Create a copy of report `/IWBEP/R_MGW_PUSH_TEST`.
2. Set a value of your model group for variable `LC_TECHNICAL_GROUP_NAME`.
3. Set a name of the collection you have created the subscription for in variable `LC_FLIGHT_COLLECTION`.
4. Save and activate your report.
5. Execute your report.

Back to [Configurations on the SAP Backend System for Pull Oriented Scenarios](#)

Related Information

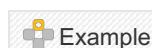
[Enabling Push/Pull for an Existing OData Channel Application](#)

[Subscription and Notification Flow for Push Oriented Scenarios](#)

1.1.4.21.1.2 Notification Content Publisher

Use

With SAP Gateway, you have the provision to write a custom code to publish the notifications.



The default content publisher provided with SAP Gateway does not support OAuth for authentication. You can write your own publisher, which supports OAuth based authentication for your application.

To custom code the content publisher, proceed as follows:

1. In transaction `SPRO` open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway** > **OData Channel** > **Outbound Flow** > **Content Publisher** > **OData Channel: Notification Content Publisher** and click the **Activity** icon to implement the BAdI `/IWFND/BD_MGW_NOTIF_SENDER`.
2. Enter a value for the BAdI filter name `DELIVERY_CHANNEL`. This is a mandatory parameter. The table below shows example of delivery channel and the corresponding delivery address.

Delivery Address	Operator	DELIVERY_CHANNEL	Description
urn:sap-com:channel:TEASNDCHANNEL/inbox	CS	TEASNDCHANNEL	The Channel Name <code>TEASNDCHANNEL</code> is the name of the RFC destination type "G" created in <code>SM59</code>
http://127.0.0.1:8080/inbox	CS	TEASNDCHANNEL	The Channel Name <code>TEASNDCHANNEL</code> is the name of the RFC destination type "G" created in <code>SM59</code> with Target host as 127.0.0.1/inbox and service as 8080

3. Implement the BAdI interface `/IWFND/IF_MGW_NOTIF_SENDER` method `SEND`. This method is passed with Outbound Notification of type `/IWFND/IF_MGW_NOTIF_TYPES=>TY_S_OUTBOUND_NOTIFICATION`. The field `xml_entry_data` in outbound notification contains the notification entity in `JSON` or `ATOM+xml` format.

Users can subscribe to a collection with delivery address. Here is the format of the delivery address:

- `urn:sap-com:channel:<Delivery Channel>/<request URI>` - where `<Delivery Channel>` is the name given to the RFC destination type "G" created in ABAP application server using transaction `SM59`.
- `http://<host>:<port>/<request URI>`

Note

1. The delivery address is optional in case the notifications are to be pulled from the SAP Gateway system.
2. If no content publisher is defined for a `DELIVERY_CHANNEL` then default Point-Point publisher - `/IWFND/CL_MGW_NOTIF_P2P_SENDER` is used for publishing the notifications.

Administrators should maintain an RFC destination type `G` with the RFC name similar to the one specified in the BAdI filter `DELIVERY_CHANNEL`.

Example

If the `DELIVERY_CHANNEL` is `TEASNDCHANNEL`, then while maintaining an RFC destination:

- RFC Destination Name is `TEASNDCHANNEL`
- Connection Type is `G`

Related Information

[Subscription and Notification Flow for Push Oriented Scenarios](#)

[Subscription and Notification Flow for Pull Oriented Scenarios](#)

[Notification Content Formatter](#)

1.1.4.21.1.3 Notification Content Formatter

Use

It is possible to format the notification payload before it gets published to the users.

Example

You can replace the internal SAP names in the payload with application specific external names.

To format the content notification, proceed as follows:

1. In transaction `SPRO` open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway** > **OData Channel** > **Outbound Flow** > **Content Formatter** > **OData Channel: Notification Content Formatter** and click the **Activity** icon to implement the BAdI `/IWFND/BD_MGW_NOTIF_FORMATTER`.
2. Enter a value for the BAdI filter name `DELIVERY_CHANNEL`. This is a mandatory parameter.
3. Implement the BAdI interface `/IWFND/IF_MGW_NOTIF_FORMATTER` method `FORMAT`. This method allows consumers to format:
 - Notification Atom Title
 - Notification Payload (Passed as deserialized ABAP type)
 - Notification End User Name
 - Notification Delivery Address
 - Notification Change Type
 - Notification Target Recipient
 - Notification HTTP Header - X-SAP-Poke-Data

Users can subscribe to a collection with delivery address. Here is the format of the delivery address:

- `urn:sap-com:channel:<Delivery Channel>/<request URI>` - where `<Delivery Channel>` is the name given to the RFC destination type "G"

- created in ABAP application server using transaction SM59.
- `http://<host>:<port>/<request URI>`

Note

The delivery address is optional in case the notifications are to be pulled from the SAP Gateway system.

Administrators should maintain an RFC destination type **G** with the RFC name similar to the one specified in the BADl filter `DELIVERY_CHANNEL`.



Example

If the `DELIVERY_CHANNEL` is `TEASNDCHANNEL`, then while maintaining an RFC destination:

- RFC Destination Name is **TEASNDCHANNEL**
- Connection Type is **G**

More Information

[OData Channel - Subscription and Notification Flow for Push Oriented Scenarios](#)

[OData Channel - Subscription and Notification Flow for Pull Oriented Scenarios](#)

[Notification Content Publisher](#)

Related Information

[Subscription and Notification Flow for Push Oriented Scenarios](#)

[Subscription and Notification Flow for Pull Oriented Scenarios](#)

[Notification Content Publisher](#)

1.1.4.21.2 Subscription and Notification Flow for Pull Oriented Scenarios

Use

Users can subscribe to get notified on any changes done to the collections. The notifications get pushed from SAP Backend to the SAP Gateway system, when any changes are made to the subscribed collection. These notifications are stored in the SAP Gateway system. Users can then pull the notifications from the SAP Gateway system whenever required using `HTTP(s) GET`.

The following topics are covered:

- [Creating a new OData Channel application for Pulling notifications](#)
- [Enabling pull for an existing OData Channel application](#)

1.1.4.21.2.1 Creating a New OData Channel Application for Pulling Notifications

Use

To create a new OData Channel pull application, you have to carry out different steps in the:

- [SAP Backend System](#)
- [SAP Gateway System](#)

1.1.4.21.2.1.1 Configurations on the SAP Backend System for Pull Oriented Scenarios

Use

Prerequisites

Make sure the following prerequisites are fulfilled:

- The software component `SAP_GWFND` is installed in your system.
- An RFC destination from the SAP backend system to the SAP Gateway system has been created with transaction `SM59`. Provide a service user and a password for the destination.
For more information, see [Maintaining Outbound bgRFC Queue from SAP Backend System to the Hub System](#).
- An SAP system alias for the SAP Gateway system has been created in the backend system.
For this an IMG activity is available in the system: In transaction `SPRO` open the SAP Reference IMG and navigate to ► **SAP NetWeaver** ► **SAP Gateway Service Enablement** ► **Backend OData Channel** ► **Connection Settings to SAP Gateway** ► **SAP Gateway Settings** .

Defining the Metadata

1. Create a class for metadata definition.
2. Inherit from the SAP Gateway object/meta model super class `/IWBEP/CL_MGW_PUSH_ABS_MODEL`.
3. Redefine method `DEFINE()` in your class to create your own implementation,
4. Add `SUPER->DEFINE()` as first statement to your implementation to define the subscription data for your application.
5. Implement metadata definition.

6. Call method `SET_SUBSCRIBABLE()` for each entity type in your implementation for which you want subscription and notification to be enabled.

Example

Class `/IWBEF/CL_MGW_MED_SFLIGHT` is delivered as an example implementation of a Subscription/Notification metadata definition.

Creating a Data Provider Class

1. Create a class for data provisioning.
2. Inherit from the SAP Gateway data provider super class `/IWBEF/CL_MGW_PUSH_ABS_DATA`.
3. Redefine abstract method `CHECK_SUBSCRIPTION_AUTHORITY` in your class to implement your checks for subscription authorization.
4. Redefine methods of interface `/IWBEF/IF_MGW_APPL_SRV_RUNTIME` in your class to process the operations of your application, for example creation or deletion of an entity.
5. Implement the data provider.

Example

Class `/IWBEF/CL_MGW_RT_SFLIGHT` is delivered as an example implementation of a subscription/notification data provider.

Sending Notifications from Backend to Consumer via SAP Gateway

1. Call static method `/IWBEF/CL_MGW_SUB_REGISTRY=>GET_SUBSCRIPTION_REGISTRY()` to get the instance of the subscription registry.
2. Call method `QUERY_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 `/IWBEF/CL_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.

Example

In the system the ABAP report `/IWBEF/R_MGW_PUSH_TEST` is available to demonstrate the notification transmission.

Continue with [Configurations on the SAP Gateway System for Pull Oriented Scenarios](#).

Configurations on the SAP Gateway System for Pull Oriented Scenarios

Use

Settings on SAP Gateway system include:

- Subscribing for Storing Notifications in the SAP Gateway System
- Pulling the Notification Stored in the SAP Gateway System

Go through the section OData for SAP Products below, before you could proceed with the SAP Gateway system settings.

OData for SAP Products - Standard Mode

The service `NOTIFICATIONSTORE` is generated under the following ICF node:

`/sap/opu/odata/iwfn/NOTIFICATIONSTORE` - Services generated under the ICF node `odata` supports notification pull only on those services, which were configured under ICF node `odata`.

Prerequisites

Make sure that you have the service 'NOTIFICATIONSTORE' activated on your SAP Gateway Hub system.

Subscribing for Storing Notifications in the SAP Gateway System

Standard Mode

Below is the service document URL:

<http://<hostname>:<port>/sap/opu/odata/iwfn/RMTSAMPLEFLIGHT/SubscriptionCollection> 

Example

In this example, the user has subscribed to changes on `FlightCollection`. But the notifications reside in the SAP Gateway system and pulled by consumers when required.

Sample code for OData for SAP Products - Standard Mode

```
Request
POST /iwfn/RMTSAMPLEFLIGHT/SubscriptionCollection
HTTP/1.1Host: <host>:<port>
accept: application/atom+xmlcontent-type:
application/atom+xmlContent-Length: 634
X-Requested-With: XMLHttpRequest

<entry xml:base="http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/" xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/2004/10/ETW/2004/10/ETW"
<id>http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT/SubscriptionCollection('005056A509B31ED2849EFB180361F2D6')</id>
<title type="text">My sub</title>
<updated>2012-10-08T03:25:45Z</updated>
<author>
<name>SAPUSER</name>
```

```

</author>
<category term="RMTSAMPLEFLIGHT.Subscription" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<link href="SubscriptionCollection('005056A509B31ED2849EFB180361F2D6')" rel="edit" title="Subscription"/>
<content type="application/xml"/>
<m:properties>
<d:ID>1234</d:ID>
<d:deliveryAddress>HTTP://10.xx.xxx.000:xxxx</d:deliveryAddress>
<d:persistNotifications>false</d:persistNotifications>
<d:collection>FlightCollection</d:collection>
<d:filter/>
<d:select/>
<d:changeType>created</d:changeType>
</m:properties>
</content>
</entry>

```

Pulling the Notification Stored in the SAP Gateway System

To pull the notifications that are stored in SAP Gateway system perform a HTTP(s) GET on entity set NotificationCollection defined in service document NOTIFICATIONSTORE. The service document, NOTIFICATIONSTORE is enhanced to support few other functionalities, see, [Pulling the Notifications Stored - Enhancements](#) for more information.

Standard Mode

Below is the Service document URL:

[http://<hostname>:<port>/sap/opu/odata/iwfnd/NOTIFICATIONSTORE/NotificationCollection?\\$format=xmlsap-client=<clientnumber>](http://<hostname>:<port>/sap/opu/odata/iwfnd/NOTIFICATIONSTORE/NotificationCollection?$format=xmlsap-client=<clientnumber>) ➦

Example

To retrieve all notifications that were created after 2011-10-03T10:00:00Z

Sample code for OData for SAP Products - Standard Mode

```

Request
GET /IWFND/NOTIFICATIONSTORE/NotificationCollection?$filter=Updated eq datetime'2013-01-28T10:18:00' HTTP/1.1
Host: <host>:<port>
accept: application/atom+xml,application/xml

Response
HTTP/1.1 200 OK
Content-Length: 658
Last-Modified: Mon, 03 Oct 2010 21:10:15 GMT
Content-Type: application/xml;type=feed;charset=utf-8
Content-Encoding:gzip;
dataserviceversion:2.0;

<?xml version="1.0" encoding="utf-8" ?>
<feed xml:base="http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/" xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" >
<id>http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection?$filter=Updated eq datetime'2013-01-28T10:18:00'</id>
<title type="text">NotificationCollection</title>
<updated>2013-03-07T09:36:34Z</updated>
<author>
</name>
</author>
<link href="NotificationCollection" rel="self" title="NotificationCollection" />
<entry>
<id>http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection('00163EA725801ED29AA67FC7B03355A6')</id>
<title type="text">NotificationCollection('00163EA725801ED29AA67FC7B03355A6')</title>
<updated>2013-03-07T09:36:34Z</updated>
<category term="NOTIFICATIONSTORE.Notification" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" />
<link href="NotificationCollection('00163EA725801ED29AA67FC7B03355A6')" rel="self" title="Notification" />
<link href="http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT;mo/FlightCollection(SAP__Origin='LOCAL',carrid='AA',connid='AA')</link>
<content type="application/xml">
<m:properties>
<d:NotificationID>00163EA725801ED29AA67FC7B03355A6</d:NotificationID>
<d:Recipient>SAPUSER</d:Recipient>
<d:ChangeType>created</d:ChangeType>
<d:Title>CUD OF FLIGHT E</d:Title>
<d:SAPOrigin>LOCAL</d:SAPOrigin>
<d:TechnicalServiceID>ZRMTSAMPLEFLIGHT_0001</d:TechnicalServiceID>
<d:Collection>FlightCollection</d:Collection>
<d:Updated>2013-01-28T10:18:00</d:Updated>
<d:ExternalServiceName>RMTSAMPLEFLIGHT</d:ExternalServiceName>
<d:EntriesOfInterest>0</d:EntriesOfInterest>
</m:properties>
</content>
</entry>
<link rel="delta" href="NotificationCollection?$filter=Updated eq datetime'2013-01-28T10:18:00'&!deltatoken='20130307093634'" />
</feed>

```


Note

Only the logged in users notifications are retrieved.

[Back to Configurations on the SAP Backend System for Pull Oriented Scenarios.](#)

1.1.4.21.2.1.2.1 Pulling the Notifications Stored - Enhancements

Use

The service document, NOTIFICATIONSTORE is enhanced to store all the notifications irrespective of whether persistent flag is set while subscribing or not. It also supports Delta Token Handling and Server Side Paging.

Delta Token Handling

With Delta token handling in NOTIFICATIONSTORE, when a query operation is performed, the user gets notifications which contains only the updated information as compared to the previous call. To get the updated information (delta), you should pass the delta link in the subsequent query.

The data returned by the feed for the query would look exactly like any other dataset for an OData query but with delta link at the end. Use this delta link to get the updated information. Here is a sample feed.

Note

Currently, deleted entries are not supported in delta token handling.

```
<feed xml:base="http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/">
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection</id>
<title type="text">NotificationCollection</title>
<updated>2012-09-10T12:50:38Z</updated>
<author>
<name/>
</author>
<link href= "NotificationCollection" rel="self" title="NotificationCollection"/>
<entry>
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection('005056A509B31EE1BEE58CE5A1C90DB0')</id>
<title type="text">CUD OF FLIGHT E</title><updated>2012-09-07T09:59:37Z</updated><category term="NOTIFICATIONSTORE.Notification"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<link href= "NotificationCollection('005056A509B31EE1BEE58CE5A1C90DB0') " rel="self" title="Notification"/>
<link href= "http://<host>:<port>/sap/opu/odata/IWFND/RMTSAMPLEFLIGHT;
mo/FlightCollection(SAP__Origin='LOCAL',carrid='AA',connid='0017',
fldate=datetime'2012-02-15T00%3A00%3A00') " rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/via" type="applicat:
title="via"/>
<content type="application/xml">
<m:properties>
<d:NotificationID>005056A509B31EE1BEE58CE5A1C90DB0</d:NotificationID>
<d:Recipient>USER</d:Recipient>
<d:ChangeType>Created</d:ChangeType>
<d:SAPOrigin>LOCAL</d:SAPOrigin>
<d:TechnicalServiceID>/IWFND/SG_SAMPLE_RMT_SFLIGHT_0001</d:TechnicalServiceID>
<d:Collection>FlightCollection</d:Collection>
<d:Updated>2013-01-28T10:18:00</d:Updated>
<d:ExternalServiceName>RMTSAMPLEFLIGHT</d:ExternalServiceName>
<d:EntriesOfInterest>0</d:EntriesOfInterest>
</m:properties>
</content>
</entry>
<link rel="delta" href="NotificationCollection?$filter=Recipient eq '*'&!deltatoken='20120910125038' " />
</feed>
```

Server Side Paging

Server side paging ensures that when a user does a query, a limited number of records as per max paging concept and a link to navigate to the next set of entries is sent. With server side paging, data provider can decide to limit the amount of data which is sent back to the client in order to ensure performance. Server-side paging is achieved by setting a skip token.

The data returned by the feed for the query would look exactly like any other dataset for an OData query but with skip token link at the end. Here is a sample feed.

```
<feed xml:base="http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/"
xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection</id>
<title type="text">NotificationCollection</title>
<updated>2012-11-28T06:32:42Z</updated>
<author>
<name/>
</author>
```

```

<link href= "NotificationCollection" rel="self" title="NotificationCollection"/>
<entry>
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection('005056A509B31ED28BC06FB04AEE9986')</id>
<title type="text">TEST USER PUSH E</title>
<updated>2012-11-14T02:32:10Z</updated>
<category term="NOTIFICATIONSTORE.Notification" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<link href= "NotificationCollection('005056A509B31ED28BC06FB04AEE9986')" rel="self" title="Notification"/>
<link href= "http://<host>:<port>/sap/opu/odata/sap/ZPUSHDEMOSRV;
mo/UserCollection(SAP__Origin='LOCAL',UserName='AAA')" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/via"
type="application/atom+xml;type=entry" title="via"/>
<content type="application/xml">
<m:properties>
<d:NotificationID>005056A509B31ED28BC06FB04AEE9986</d:NotificationID>
<d:Recipient>USER</d:Recipient>
<d:ChangeType>Created</d:ChangeType>
<d:SAPOrigin>LOCAL</d:SAPOrigin>
<d:TechnicalServiceID>ZPUSHDEMOSRV_0001</d:TechnicalServiceID>
<d:Collection>UserCollection</d:Collection>
<d:ExternalServiceName>ZPUSHDEMOSRV</d:ExternalServiceName>
<d:Updated>2013-01-28T10:18:00</d:Updated>
<d:EntriesOfInterest>0</d:EntriesOfInterest>
</m:properties>
</content>
</entry>
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
<link rel="next" href="NotificationCollection?$skiptoken=71"/>
</feed>

```

Delta Token Handling with Server Side Paging

It is important that the server calculates the delta token the moment a query is executed, considering the instance data could change while a client is paging through results using a series of next links. This allows the client to use a delta link to fetch the changes occurred for a set of data since the query was first executed. This is achieved by holding the delta token in the next link.

The data returned by the feed for the query would look exactly like any other dataset for an OData query but with a delta token and a skip token link at the end. Here is a sample feed.

```

<feed xml:base="http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/"
xmlns="http://www.w3.org/2005/Atom" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices">
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection?!deltatoken='20120928063142'</id>
<title type="text">NotificationCollection</title>
<updated>2012-11-28T06:35:14Z</updated>
<author>
<name/>
</author>
<link href= "NotificationCollection" rel="self" title="NotificationCollection"/>
<entry>
<id> http://<host>:<port>/sap/opu/odata/IWFND/NOTIFICATIONSTORE/NotificationCollection('005056A509B31ED28BA74845D8A54DE5')</id>
<title type="text">TEST USER PUSH E</title>
<updated>2012-11-13T02:31:41Z</updated>
<category term="NOTIFICATIONSTORE.Notification" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<link href= "NotificationCollection('005056A509B31ED28BA74845D8A54DE5')" rel="self" title="Notification"/>
<link href= "http://<host>:<port>/sap/opu/odata/sap/ZPUSHDEMOSRV;
mo/UserCollection(SAP__Origin='LOCAL',UserName='AAA')" rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/via"
type="application/atom+xml;type=entry" title="via"/>
<content type="application/xml">
<m:properties>
<d:NotificationID>005056A509B31ED28BA74845D8A54DE5</d:NotificationID>
<d:Recipient>USER</d:Recipient>
<d:ChangeType>Created</d:ChangeType>
<d:SAPOrigin>LOCAL</d:SAPOrigin>
<d:TechnicalServiceID>ZPUSHDEMOSRV_0001</d:TechnicalServiceID>
<d:Collection>UserCollection</d:Collection>
<d:Updated>2013-01-28T10:18:00</d:Updated>
<d:ExternalServiceName>ZPUSHDEMOSRV</d:ExternalServiceName>
<d:EntriesOfInterest>0</d:EntriesOfInterest>
</m:properties>
</content>
</entry>
+<entry>
+<entry>
+<entry>
+<entry>

```

```
+<entry>
+<entry>
+<entry>
+<entry>
+<entry>
<link rel="next" href= "NotificationCollection?!deltatoken='20120928063142'&$skiptoken=11">
</feed>
```

1.1.4.21.3 Enabling Push/Pull for an Existing OData Channel Application

Use

If you have already implemented an OData Channel application in your system, you can later specify this for push/pull functionality.

Prerequisites

Make sure the following prerequisites are fulfilled:

- The software component `SAP_GWFND` is installed in your backend system.
- You have already created an OData Channel application and you want to enable the push/pull functionality.

Activities

1. Open your OData Channel metadata and data provider classes in `SE24`.
2. On tab `Properties` choose `Change Inheritance`.
3. Specify that you want to keep your existing implementations of redefined methods.
4. Continue with the implementation as described in.
 - [Creating a New OData Channel Push Application](#)
 - [Creating a New OData Channel Pull Application](#)
5. Activate and save your class.

1.1.4.21.4 Maintaining Outbound bgRFC Queue from SAP Backend System to the Hub System

Use

To use the bgRFC, the following configuration steps are required:

- Creating RFC destination for outbound queue
- Registering RFC destination for outbound queue

Creating RFC Destination for Outbound Queue

1. Go to transaction `SM59` in the SAP backend system.
2. Choose `Create` to create a new connection.
3. In the `RFC Destination` field, enter the name of the RFC destination.
4. In the `Connection Type` field, enter `3 (Connection to ABAP System)`.
5. Select the `Special Options` tab and choose the `qRFC Version` as bgRFC (or) in higher releases choose the `Transfer Protocol as 1 Classic with bgRFC`.
6. Save the RFC destination and choose `Connection Test` to test the connection.

Registering RFC Destination for Outbound Queue

1. Go to transaction `SBGRFCCONF` in the SAP Gateway system.
2. Go to the `Define Inbound Dest.` tab and choose `Create`.
3. In the `Inb. Dest. Name` field, enter the name of the RFC destination.
4. Choose `Save` to save the queue prefixes.
5. Go to the `Scheduler: Destination` tab and choose `Create`.
6. In the dialog box, select Inbound as the destination type.
7. In the `Destination` field, enter the name of the RFC destination.
8. Choose `Save` to save the scheduler setting you have just created.
9. In the `bgRFC Destination` screen, choose `Save` to save your customizing settings.

Related Information

[Creating a New OData Channel Application for Sending Notifications](#)

[Creating a New OData Channel Application for Pulling Notifications](#)

1.1.4.21.5 Maintaining Inbound bgRFC Queue on the Hub System

Use

SAP Gateway uses inbound queues to reliably send information to a consumer (Such queues are used by various services. For example, the [Notification Content Publisher](#). These inbound queues use bgRFC (Background Remote Function Call) technology.

To use the bgRFC, the following configuration steps are required:

- Create the RFC connection `IWFND_ODATA_PUSH` for the inbound queues.
- Register the RFC connection `IWFND_ODATA_PUSH` for the inbound queues

Creating RFC Destination for Inbound Queue

1. Go to transaction `SM59` in the SAP Gateway system.
2. Choose **Create** to create a new connection.
3. In the **RFC Destination** field, enter `IWFND_ODATA_PUSH`.
4. In the **Connection Type** field, enter `3` (Connection to ABAP System).
5. Select the **Special Options** tab and choose the **qRFC Version** as bgRFC (or) in higher releases choose the **Transfer Protocol as 1 Classic with bgRFC**.
6. Save the RFC destination and choose **Connection Test** to test the connection.

Registering RFC Destinations for Inbound Queue

1. Go to transaction `SBGRFCCONF` in the SAP Gateway system.
2. Go to the **Define Inbound Dest.** tab and choose **Create**.
3. In the **Inb. Dest. Name** field, enter `IWFND_ODATA_PUSH`.
4. Choose **Save** to save the queue prefixes.
5. Go to the **Scheduler: Destination** tab and choose **Create**.
6. In the dialog box, select Inbound as the destination type.
7. In the **Destination** field, enter `IWFND_ODATA_PUSH`.
8. Choose **Save** to save the scheduler setting you have just created.
9. In the **bgRFC Destination** screen, choose **Save** to save your customizing settings.

Related Information

[Creating a New OData Channel Application for Sending Notifications](#)

1.1.4.21.6 Subscription Management

Use

Note

This service will support creation of subscriptions only in the context of the new OData library 1.0.

An administrator can subscribe for notifications on behalf of a user or a role (a set of users). In both the cases, the users subscribed by the administrator gets the notifications and not the administrator. The notifications for the subscribed collections are pushed from the SAP backend to the SAP Gateway Hub system when the collections are modified. SAP Gateway then pushes the notifications using `HTTP POST` to all subscribers based on their delivery address. In case of roles, they are resolved by application specific implementation in the SAP Gateway system and corresponding users are sent notifications.

Following topics are covered:

- Operations Supported
- Authorization Details
- [An example to Create Subscription on Behalf of a User](#)
- [Fetching the list of users for a role](#)
- [Multiple Origin Composition \(MOC\) for Subscription Management](#)

Service Document

Here is the link to access the service document URL for subscription management:

`http://<hostname>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT/?$format=xml`

Note

Contact your system administrator for host name and port number.

Operations Supported

An administrator can:

- Create subscription for any user or role
- Update subscription for any user or role
- Delete any subscription either created by the administrator or any user
- Read and Query any number of subscriptions provided which is in the context of new OData library 1.0

Assigning Data Provider to Data Model

Note

- Proceed with the following configuration if you are using the original version (version 1) of `SUBSCRIPTIONMANAGEMENT` service and you want it to **support content based routing**. If you are using version 2 of `SUBSCRIPTIONMANAGEMENT` service, the following configuration is not applicable.
- Version 2 of `SUBSCRIPTIONMANAGEMENT` service will be active if earlier version (version 1) is already available in active state. You only need to activate the version 2 of `SUBSCRIPTIONMANAGEMENT` service if it is not already in the active state. See [Activate and Maintain Services](#) for more information.

1. In transaction SPRO open the SAP Reference IMG and navigate to: ► SAP NetWeaver ► SAP Gateway ► OData Channel ► Registration ► Assign Data Provider to Data Model and click on the Activity icon.
2. Choose the Model Identifier you created while activating the version 1 of SUBSCRIPTIONMANAGEMENT service. For example, ZOM_MGW_ADM_SUB_MODEL_0001_BE.
3. Change the Class/Interface to /IWFND/CL_MGW_ADP_VIRTUAL_PERS.
4. Save your settings.

Authorization Details

The user requires specific authorization for the services in the system through the object /IWBEP/SUB. This is to limit the access to specific Business areas. The Role Template /IWBEP/RT_SUB_USR is used to create roles with the required authorization. The authorization checks are performed in two steps:

Level	Authorization Check	Description	Field	Value
1	Authorization check for SUBSCRIPTIONMANAGEMENT service	This is the first level of check where the authorization to the administrator to create subscriptions on behalf of users or roles is checked.	/IWBEP/GTN	/IWBEP/SUBSCRIPTIONMANAGEMENT
ACTVT	* (all operations)			
2	Authorization check for specific services	This is the second level check where the authorization to create subscription for a service is checked. For example, If the Administrator wants to create subscriptions on behalf of a user for the service RMTSAMPLEFLIGHT, the administrator needs the authorization mentioned here in level 2 and level 1.	/IWBEP/GTN	/IWBEP/RMTSAMPLEFLIGHT
ACTVT	* (all operations)			

1.1.4.21.6.1 An Example to Create Subscription on Behalf of Users

Use

<p>Request</p> <pre>POST /IWBEP/SUBSCRIPTIONMANAGEMENT/SubscriptionCollection HTTP/1.1 HOST: <host>:<port> accept: application/atom+xml content-type: application/atom+xml Content-Length: 634 X-Requested-With: XMLHttpRequest X-CSRF-Token: krVmAnoj48a0mn36shLKRA== <?xml version="1.0" encoding="utf-8"?> <atom:entry xml:base="http://<host>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT/" xmlns:atom="http://www.w3.org/2005/Atom" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"> <atom:category term="SUBSCRIPTIONMANAGEMENT.Subscription" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/> <atom:content type="application/xml"> <m:properties> <d:CreatedBy/> <d:TechnicalServiceName>/IWRK/WFODCPROCESSING</d:TechnicalServiceName> <d:TechnicalServiceVersion>0001</d:TechnicalServiceVersion> <d:DeliveryAddress>http://XXX.XXX.1.1:XXXX/alerts/</d:DeliveryAddress> <d:PersistNotifications>false</d:PersistNotifications> <d:Collection>WorkflowTaskCollection</d:Collection> <d:Filter/> <d>Select>*</d>Select> <d:ChangeType>updated</d:ChangeType> <d:Principal>SAPUSER</d:Principal> <d:PrincipalType>USER</d:PrincipalType> <d:BaseUrl/> </m:properties> </atom:content> </atom:entry></pre>	<p>Response</p> <pre>HTTP/1.1 201 Created Content-Length: 650 Content-Type: application/xml;type=entry Content-Encoding:gzip</pre>
--	--

```
dataserviceversion:2.0

<?xml version="1.0" encoding="utf-8"?>
<atom:entry xml:base="http://<host>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT/" xmlns:atom=
"http://www.w3.org/2005/Atom" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices" xmlns
:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata">
<atom:category term="SUBSCRIPTIONMANAGEMENT.Subscription" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/>
<atom:content type="application/xml">
<m:properties>
<d:ID>005056A509D41EE194DF0F559163A0DA</d:ID>
<d:CreatedBy>SAPUSER</d:CreatedBy>
<d:TechnicalServiceName>/IWRK/WFODCPROCESSING</d:TechnicalServiceName>
<d:TechnicalServiceVersion>0001</d:TechnicalServiceVersion>
<d:DeliveryAddress>http://XXX.XXX.1.1:XXXX/alerts/</d:DeliveryAddress>
<d:PersistNotifications>false</d:PersistNotifications>
<d:Collection>WorkflowTaskCollection</d:Collection>
<d:Filter/>
<d>Select>*</d>Select>
<d:ChangeType>updated</d:ChangeType>
<d:Principal>SAPUSER</d:Principal>
<d:PrincipalType>USER</d:PrincipalType>
<d:BaseUrl>http://<host>:<port>/sap/opu/odata/IWCNT/WFODCPROCESSINGhttp://<host>:<port>/sap/opu/odata/IWCNT/WFODCPROCESSING</d:Bas
</m:properties>
</atom:content>
<atom:id>http://<host>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT/SubscriptionCollection('005056A509D41EE194DF0F559163A0DA')
<atom:link href="SubscriptionCollection('005056A509D41EE194DF0F559163A0DA') " rel="edit" title="Subscription"/>
<atom:title type="text">SubscriptionCollection('005056A509D41EE194DF0F559163A0DA')</atom:title>
<atom:updated>2012-02-09T08:31:36Z</atom:updated>
</atom:entry>
```

Similarly, subscription for a Role is created by passing the below parameters:

```
<d:Principal>SAP_ROLE</d:Principal>
<d:PrincipalType>ROLE</d:PrincipalType>
```

Note

As seen in the above example, the base URL is passed as request. Hence, the base URL is constructed from the Administrator's service context and the external name of the service.

1.1.4.21.6.2 Fetching the List of Users for Roles

Use

Note

Role to user resolution is an additional capability provided and you can use it if required.

The role to user resolution can either be on the SAP backend system or on the SAP Gateway Hub system. The method `GET_USERS_FROM_ROLES` of the API `/IWBEP/CL_MGW_PUSH_UTIL` on the SAP backend system returns the list of users for a role or a list of roles.

- In case the role user mapping is on the SAP Gateway Hub system, implement the BAdI `/IWBEP/BD_MGW_PUSH_ROL_MAP_PRV` and use the class `/IWBEP/CL_MGW_PUSH_ROL_MAP_RMT` as the implementation.
The BAdI is executed by the `GET_USERS_FROM_ROLE` method of the utility class `/IWBEP/CL_MGW_PUSH_UTIL` for fetching the list of users for a role or list of roles and the appropriate system ID and system client of the SAP Gateway Hub system must be provided.
- In case the role user mapping is on the SAP backend system, then the optional parameters `IV_SOURCE_SYSTEM_ID` and `IV_SOURCE_CLIENT` can be ignored.
The list of roles should be passed as input in the table `IT_ROLE_TAB`.

Utility Class for Push Notifications

The interface `/IWBEP/CL_MGW_PUSH_UTIL` on the SAP backend system returns the list of users for a role or the list of roles.

Method `GET_USERS_FROM_ROLES` gets the list of users corresponding to a role.

Parameter	Description
<code>IV_SOURCE_SYSTEM_ID</code>	The system ID from which, role to user mapping is fetched
<code>IV_SOURCE_CLIENT</code>	The system client from which, role to user mapping is fetched
<code>IT_ROLE_TAB</code>	The list of roles for which, the assigned users are fetched
<code>ET_ROLE_USER_TAB</code>	The list of users for the roles passed in <code>ET_ROLE_USER_TAB</code>

1.1.4.21.6.3 MOC for Subscription Management

Use

Multiple Origin Composition (MOC) is the ability to collect data from different backend systems, aggregate them in one single service and updating different backend systems while using the same user. This is now extended to subscription management where, an administrator can configure subscriptions on behalf of a user in multiple backend systems. This means that the administrator can create, read, update, delete, and query subscriptions across multiple systems.

Prerequisites

- Make sure that you have `SUBSCRIPTIONMANAGEMENT` OData service activated on your SAP Gateway Hub system with more than one SAP backend system configured for it and with one of them set as default.
- Make sure that the other OData services you have subscribed for is activated on your SAP Gateway Hub system with more than one backend system configured for it and with one of them set as default.
- The system alias configured for the service that can be subscribed for (target service) should be a subset of the system alias configured for the `SUBSCRIPTIONMANAGEMENT` service.

SUBSCRIPTIONMANAGEMENT - System Alias	Default	RMTSAMPLEFLIGHT - System Alias	Default
A	X	B	X
B		C	
C			

Assumptions

- The OData service `SUBSCRIPTIONMANAGEMENT` should be configured against all those backend systems where administrator would like to subscribe a user to an OData service.
- The `SUBSCRIPTIONMANAGEMENT` OData service creates a subscription entry for a service in only those backend systems where the service is configured.
- The administrator should make sure that principal types (Roles and Users) are present in all the backend systems used.
- The `SUBSCRIPTIONMANAGEMENT` OData service throws an error in case the subscription could not be created, updated, or deleted from all backend systems successfully.
- The subscriptions created by `SUBSCRIPTIONMANAGEMENT` OData service can be updated or deleted only through `SUBSCRIPTIONMANAGEMENT` service.
- When a read operation is performed on the `SUBSCRIPTIONMANAGEMENT` OData service, it returns subscriptions only from the backend system which is set as default.
- The OData service `SUBSCRIPTIONMANAGEMENT` returns a value " *" for `SAP-Origin`, indicating that subscription entries are present in more than one backend system.

Assigning Data Provider to Data Model

Note

- Proceed with the following configuration if you are using the original version (version 1) of `SUBSCRIPTIONMANAGEMENT` service and you want it to support the MOC functionality. If you are using version 2 of `SUBSCRIPTIONMANAGEMENT` service, the following configuration is not applicable.
- Version 2 of `SUBSCRIPTIONMANAGEMENT` service will be active if earlier version (version 1) is already available in active state. You only need to activate the version 2 of `SUBSCRIPTIONMANAGEMENT` service if it is not already in the active state. See [Activate and Maintain Services](#) for more information.

1. In transaction `SPRO` open the SAP Reference IMG and navigate to: **SAP NetWeaver** > **SAP Gateway** > **OData Channel** > **Registration** > **Assign Data Provider to Data Model** and click on the **Activity** icon.
2. Choose **New Entries** and enter the following:

Field	Description
Model Identifier	The technical name provided for Model Identifier while activating the version 1 of <code>SUBSCRIPTIONMANAGEMENT</code> service. For example, <code>ZOM_MGW_ADM_SUB_MODEL_0001_BE</code> .
Type	Choose the option Multi Destination Composition Data Provider .
Software Version	Choose DEFAULT .
Class/Interface	Enter <code>/IWFND/CL_MGW_MDC_VIRTUAL_PERS</code> .
Description	Enter a description.

3. Save your settings.

Service Document

Here is the link to access the service document URL:

[http://<hostname>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT;mo/?\\$format=xml](http://<hostname>:<port>/sap/opu/odata/IWBEP/SUBSCRIPTIONMANAGEMENT;mo/?$format=xml)

Note

Contact your system administrator for host name and port number.

Related Information

[Multiple Origin Composition](#)

1.1.4.22 Namespace Handling in Model Provider Class

You can set the namespace alias for an annotation target with method `/IWBEP/IF_MGW_VOCAN_ANN_TARGET~SET_NAMESPACE_QUALIFIER`.

For handling of the schema namespace by the application the following API method is available:
/IWBEP/IF_MGW_ODATA_MODEL~SET_SCHEMA_NAMESPACE.

Example

```
model->set_schema_namespace('Tea_test_application_NS').
```

The schema namespace appears in the metadata document as follows:

```
1. <Schema Namespace="Tea_test_application_NS" xml:lang="en"..
2. <Property Name="Location" Type="Tea_test_application_NS.CT_Location" Nullable="false" />
3. <NavigationProperty Name="My_Team"
Relationship="Tea_test_application_NS.Team_of_Employee"
FromRole="FromRole_Team_of_Employee" ToRole="ToRole_Team_of_Employee" />
4. <Association Name="Team_of_Employee" sap:content-version="1">
<End Type="Tea_test_application_NS.Worker" Multiplicity="*"
Role="FromRole_Team_of_Employee" />
<End Type="Tea_test_application_NS.Team" Multiplicity="1"
Role="ToRole_Team_of_Employee" />
5. <EntityContainer Name="Tea_test_application_NS.Entities"
m:IsDefaultEntityContainer="true"/>
6. <ComplexType Name="CT_Location">
<Property Name="Country" Type="Edm.String" MaxLength="255" sap:text="Country" />
<Property Name="City" Type="Tea_test_application_NS.CT_City" Nullable="false" />
<Property Name="Complex_type_City" Type="Tea_test_application_NS.CT_City" Nullable="false" />
</ComplexType>
```

Related Information

[/IWBEP/IF_MGW_ODATA_MODEL](#)

1.1.5.2.1.17 /IWBEP/IF_MGW_EXPR_VISITOR

Use

This is a visitor interface which can be implemented to process the filter expression tree.

Methods

Method PROCESS_LITERAL

This method processes the literal expression of the filter expression tree.

Parameter	Description
IO_LITERAL	This parameter is of type /IWBEP/IF_MGW_EXPR_LITERAL which contains literal expression.

Method PROCESS_PROPERTY

This method processes the property expression of the filter expression tree.

Parameter	Description
IO_PROPERTY	This parameter is of type /IWBEP/IF_MGW_EXPR_PROPERTY which contains property expression.

Method PROCESS_UNARY

This method processes the unary expression of the filter expression tree.

Parameter	Description
IO_UNARY	This parameter is of type /IWBEP/IF_MGW_EXPR_UNARY which contains unary expression.

Method PROCESS_BINARY

This method processes the binary expression of the filter expression tree.

Parameter	Description
IO_BINARY	This parameter is of type /IWBEP/IF_MGW_EXPR_BINARY which contains binary expression.

Method PROCESS_FUNCTION

This method processes the function expression of the filter expression tree.

Parameter	Description
IO_FUNCTION	This parameter is of type /IWBEP/IF_MGW_EXPR_FUNCTION which contains function expression.

Method PROCESS_MEMBER

This method processes the member expression of the filter expression tree.

Parameter	Description
IO_MEMBER	This parameter is of type /IWBEF/IF_MGW_EXPR_MEMBER which contains member expression.

1.1.4.23 Integration with Unit Test Framework

Data provider classes are enabled for unit tests. A data provider (specialization of /IWBEF/CL_MGW_ABS_DATA) requires the request context (a realization of the /IWBEF/IF_MGW_REQ_X interfaces) to be initialized in order to receive the information provided through the consumer request.

An implementation of the request context is available with /IWBEF/CL_MGW_REQUEST_UNITTST which is a subclass of /IWBEF/CL_MGW_REQUEST. The new request context is exposed in the package interface /IWBEF/MGW_GSR_CORE_CNT, hence is accessible for the data provider classes (DPCs). In addition, the new method INIT_DP_FOR_UNIT_TEST has been added to the interface /IWBEF/IF_MGW_CONV_SRV_RUNTIME. This method is used to initialize a data provider instance for the usage in a unit test.

The process is as follows:

1. An instance of the new request context is created.
2. The request values are assigned to the request context instance.
3. Member variables of the /IWBEF/CL_MGW_ABS_DATA are initialized and set:
 - MO_CONTEXT
 - MR_REQUEST_DETAILS
 - MR_SERVICE_DOCUMENT_NAME
 - MR_SERVICE_VERSION
 - MR_SERVICE_NAMESPACE
4. Logger and message container instances are retrieved.

Note

The implementation of the request context can only be used within the unit test, not during usual processing through the SAP Gateway runtime.

Setting Up a Unit Test for a Data Provider Class

1. Create the unit test class of your DPC.
2. Declare the following variables in the definition section of your unit test class:

- The request context object
`data: mo_request_context_object type ref to /iwbep/cl_mgw_request_unittst.`
- The request context structure to be filled in the unit test class:
`data: ms_request_context_struct type /iwbep/cl_mgw_request_unittst=>ty_s_mgw_request_context_unit.`
- The data provider instance of your data provider class:
`data: mo_data_provider type ref to <your DPC>.`
- Create the data provider instance in the setup method of your unit test:

Source Code

```
method setup.  
* instantiate the Data Provider  
create object mo_data_provider .
```

3. Test your DPC methods by using test_get_entityset_managers, for example.
 1. Fill the request context structure with the values corresponding to your use case, for example navigation path, order, select, keys, entity set, etc.

```
ms_request_context_struct-technical_request-source_entity_type = 'Manager'.  
  
ms_request_context_struct-technical_request-target_entity_type = 'Manager'.  
  
ms_request_context_struct-technical_request-source_entity_set = 'Managers'.  
  
ms_request_context_struct-technical_request-target_entity_set = 'Managers'.
```

2. In order to simulate converted keys, action parameters or filters later on, they need to be set in the request context structure explicitly:

```
DATA: lo_filter TYPE REF TO /iwbep/if_mgw_req_filter.  
  
DATA: ls_converted_keys TYPE REF TO data.  
  
DATA: ls_converted_parameters TYPE REF TO data.  
  
ms_request_context_struct-technical_request-converted_keys = ls_converted_keys.  
  
ms_request_context_struct-technical_request-conv_action_parameters = ls_converted_parameters.  
  
ms_request_context_struct-technical_request-filter_object = lo_filter .
```

The variables `ls_converted_keys` and `ls_converted_parameters` have to be populated by the provider itself. It is only relevant if conversion is required for parameters at all which depends on the model. A convenience method is not available for conversion of input data.

3. Call the method below in order to initialize the data provider instance for the unit test:

```
mo_request_context_object = mo_data_provider->/iwbp/if_mgw_conv_srv_runtime~init_dp_for_unit_test ( is_re
```

4. Now you can call the data provider implementation as usual. For your convenience the `/iwbp/if_mgw_appl_srv_runtime` methods have been simplified to only import the technical request context object.

```
try .
    so_data_provider->/iwbp/if_mgw_appl_srv_runtime~get_entityset (
        exporting
            io_tech_request_context    = so_request_context_object
        importing
            er_entityset               = lr_data
            es_response_context        = ls_response_context
    ).

    catch /iwbp/cx_mgw_busi_exception.

    catch /iwbp/cx_mgw_tech_exception.

endtry.
```

5. Implement the corresponding unit test assertions and checks as required.

```
cl_aunit_assert=>assert_not_initial( act = lr_data ).
```

Example

You can find an example unit test created for the data provider class `/IWBP/CL_TEA_DATA_PROVIDER` in the system.

1.1.5 APIs and Coding

Use

In this section you can find information on programming APIs and coding:

- [Logging in SAP Gateway](#)
- [OData Channel APIs](#)
- [Debugging Query Option](#)

Logging in SAP Gateway

Use

For SAP Gateway monitoring, files are generated and saved for the **system log** and the **application log**.

- For system logs, go to transaction `SM21`.
- For application logs, go to transaction `SLG1` and open the SAP Gateway Application Log Viewer `/IWFND/`.

Logs are persisted on the ABAP database. To configure logging settings, go to the SAP Reference IMG in transaction `SPRO` and choose the activities under the following node: **SAP NetWeaver > SAP Gateway > OData Channel > Administration > Logging Settings**.

The three activities **Set Log Level for All Users**, **Set Log Level for Specific Users**, and **Set Display of Sensitive Information** allow you to specify the settings for log and trace files.

Logging API

The logging API provided by the class `/IWFND/CL_LOGGER` provides different methods to serve different use cases or circumstances. It differentiates between user/request specific and non-user/request specific situations.

Depending on the category, the log information is directed to the corresponding tool. The administrator can find user/request specific logs in the Solution Manager Diagnostics; non-user/request specific logs are in the CCMS.

Notes for `/IWFND/CL_LOGGER`

GW Agent	Describes the agent, component or class that creates the log entry, for example, Destination Finder, Consumer Connectivity, <code>/IWFND/CL_DEVICE_MANAGER</code> .
Object key and/or system alias	If both are available, both should be specified. If only the object key is available, the system alias will be derived from it. When a SAP system calls a SAP Gateway system, the system alias should contain the name of the SAP system.
Message container	If a message is logged through the message container, it must not also be logged through the Logger API, since the message container also logs its messages through the Logger API.
MSG_HANDLE	This parameter maintains the correlation between the step initialization and the step completion message in the Applications Log. The <code>MSG_HANDLE</code> parameter is returned using the method <code>LOG_STEP_INIT</code> . You must store the value internally until you pass it to the <code>LOG_STEP_COMPLETION</code> or <code>LOG_STEP_COMPLETION_EXCEPTION</code> methods.
BO items	The content of BO items must not be passed in any of the parameters.
Table <code>/IWFND/V_CCMS</code>	Recommendation for logging in Applications Log: The table <code>/IWFND/V_CCMS</code> contains values for the most of the available framework agents. If you want to pass the name of an agent, use the values maintained in this table.

Caution

Note the following **security aspects**:

- You should **not** log T100 message variables if they contain person-related, sensitive data.

Preliminary Considerations

Consider the following aspects before deciding where to log a message:

Message Type	Description
User/request related messages	These are messages issued for a specific user and/or request context.
Non-user/request related messages	These messages describe a system-wide, cross-user, component-wide or cross-system failure or other critical state. Such a situation affects the entire system, application or component.
End-user related messages	These messages describe a failure or state the end-user can influence, for example, by executing an activity in a different way or retrying a certain request with different parameters.
Administration-related messages	These messages describe the processing steps/results of a user request. The information provided in these messages can be used by the administrator or by a support person to solve problems. These messages typically contain information that is not understandable to the end user.

Notes for T100 Messages

For framework-related messages use the message class `/IWFND/CM_LOGGING` to provide a message per implementation step for your component and/or agent. Message example: Step init: Map the proxy structure to the GenIL Data Container.

For applications: use your own message class to describe application-specific steps.

In your component and/or agent specific message class, provide descriptions for the different step completion cases:

- Step was completed successfully.
For example: Mapping the proxy structure to the GenIL Data Container was completed successfully
- Step finished with a warning, error, abort.
For example: Mapping the proxy structure to the GenIL Data Container ended with an error/warning. Error description ...
Provide the long text if you have to describe the problem in detail and if you already have potential solution capabilities and options.
- Step finished with an exception.
For example: Mapping the proxy structure to the GenIL Data Container ended with an exception
The exception itself will be described through the exception object and is logged in a separate log message internally.

Logging Implementation Steps

- Initiating the logger
- Completing a logging step
- Logging an exception
- Logging BAPI returns
- Logging simple messages
- Logging simple messages through a structure
- Logger handling for outbound flows

User/Request Specific Logging in the Application Log

The following table depicts the essential steps executed by the SAP Gateway components and agents. To ensure that the processing of a request on SAP Gateway becomes transparent and understandable, these steps must be logged correctly. Therefore, it is necessary to keep the process defined after the table.

Layer	Agent	Use Case
Consumer Connectivity	Consumer Application Inbound Service	<ul style="list-style-type: none">Map the proxy structure to the GenIL Data ContainerPrepare the WS responseReturn response to the SAP GWCheck if the request ID has been processed (resumability and idempotence)
Consumer Outbound Service	<ul style="list-style-type: none">Determine relevant users (currently done in the GSI implementation)Determine relevant consumerDetermine relevant outbound channelMap the outbound document	
Common Services	Event Manager	<ul style="list-style-type: none">Registration of eventProcess registered eventsInvoke event handlerTransfer data to event handler
Event Handler	Application specific logic	
Instance Manager	<ul style="list-style-type: none">Retrieve system alias from input or destination finderRetrieve version aliases for system aliases from destination finder	

	<ul style="list-style-type: none"> • Instantiate GSI implementation(s) for an inbound request • Find composition BSO • Instantiate GSI implementation(s) for a system alias and GSDO type 	
Device Manager	<ul style="list-style-type: none"> • Create or update device object • Remove device details • Get the details of a device with the specified DEVICE_ID • Get the details of a device with the specified user id • Get the user language for a specific user id • Update a user device object according to the specified consumer name • Get the details of a GW with the specified user id and device • Get/Set application ID and the related version • Get/Set application attributes 	
Message Container		
Destination Finder	<ul style="list-style-type: none"> • Find the corresponding list of system aliases for an inbound request. • Get RFC destination for a system alias/get WS destination. • Get RFC destination for an explicit user for a system alias. • Get system alias description. 	
Cache/Persistence Manager		
Business Notification Sender		
Document Publisher	<ul style="list-style-type: none"> • Instantiate the consumer adapter class. • Call the consumer adapter's sent message. • Extract the GSDO data. • Read the document from the attachment storage. • Get front-end user via user mapping. • Call the consumer Web Service via the bgRFC wrapper function Update OIL. • Get reports from the report database for the report ID obtained from the GSDO. 	
Generic Event Handler	<ul style="list-style-type: none"> • Read the GSDO instance. • Fetch the subscriber list from the GSDO node. • Read Device Manager information for the subscriber list. • Call content publisher send method. 	
Consumer Adaptation Layer		<ul style="list-style-type: none"> • Mapping logic for flat WS. • Field conversions.
Interface Layer	Generic Service Consumer	<ul style="list-style-type: none"> • Determine the GenIL component. • Delegate the request to the GenIL runtime. • Cache side effects. • Error processing. • Event processing (notify event manager about side effects).
GenIL - no logging	<ul style="list-style-type: none"> • Instantiate the Service Provider class. • Delegate the query request to the Service Provider class. • Return DO and side effects. 	
Generic Service Provider	<ul style="list-style-type: none"> • Call the Instance Manager to instantiate the GSI implementation. • Delegate the query (+CRUD) request to the GSI implementation. • Convert the GenIL data container into a GSDO list. • Execute DB commit and initiate remote update and commit/rollback at the consumer connectivity layer. • Create preliminary ID (create method). • Publish a mapping between the preliminary ID and the permanent ID (save method). • Map enriched data from the SAP system to the GenIL DO. • Delegate to GSI check before save to request potential checks. • Request local update prior to remote update. • Remote update. • Local update after remote update. • Compensation logic. 	

	<ul style="list-style-type: none"> • Clean-up of local caches. • Request the reset of internal buffers. 	
BAL	Generic GSI	<ul style="list-style-type: none"> • Request to buffer the modification/deletion of a GSDO. • Check if the GSDOs that will be modified can be persisted. • Request DB update prior remote update. • Request remote update.
Abstract GSI	<ul style="list-style-type: none"> • Extract the IDs out of a GSDO list. • Return the BOP instance. Delegate the mapping of the selection parameters to the BOP DO. • Delegate the mapping of the result set out of the BOP DO into the GSDO list. • Delegate the mapping of the IDs out of IO_REQUEST_DATA into the BOP DO. 	
Composition BSO	Delegate requests.	
Adaptation BSOs	<ul style="list-style-type: none"> • Adapt the query result set to the canonical format, which in this example is identical to the GenIL object mode. • Insert the adapted query result into the SAP Gateway data container. • Adapt the query parameter to the SAP system format. • Delegate the query (+CRUD) to the BOP. • Cache update request for local and remote objects. • Perform local update (after remote call). • Publish modified GSDOs using the list of changed instances. • Provide permanent ID (save remote method). 	
CSSO	Mapping prior and after delegation to BSO.	
SAP System Connectivity	Backend Outbound Service	<ul style="list-style-type: none"> • For each Web Service proxy and SAP system, a logical port must be defined. • Execute the SAP system operations and/or check BADI suppression flag. • Create a BOP. • Provide a reference to the extension container of the GSDO that is provided to the BAdIs. • Provide a reference to the enhancement class BAdI.
Backend Inbound Service	<ul style="list-style-type: none"> • Process inbound message. • Call BgRFC unit. • Asynchronous processing (call BSO, cache, call event manager). • Adapt SAP system data to canonical GSDO format (optional as same as READ for workflow and reporting). 	
SAP System	SAP GW Application logic	Application-specific logic

1.1.5.1.1 Initiating the Logger

Use

Use the method `INIT_LOGGER` of class `/IWFND/CL_LOGGER` to create the logger instance and to initiate the Applications Log protocol.

- For inbound processes:
This method should be called directly after a request has reached the Connectivity Layer. It will be used only by the framework.
- For outbound processes:
See [Logger Handling for Outbound Flows](#).
- Object and/or sub-object:
For application coding you need not specify these parameters when the logger is initiated. The logger will be initialized with the object `/IWFND/` and sub-object `DESTIM` only for design time processes.
- Request direction:
Specifies the direction of the request. Possible values are maintained as constants in the class itself and in the domain `/IWFND/SUP_REQ_DIR`.
- Operation:
Specifies the operation executed on the business object type in the current request.



Example

Examples are Create, Update, Delete, and Query.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IV_USERID	importing		UNAME	User Name

IV_REQUESTGUID	importing	X	STRING	Request GUID
IV_OBJECT	importing	X	BALOBJ_D	Object name
IV_SUBOBJECT	importing	X	BALSUBOBJ	Subobject
IV_REQUEST_DIRECTION	importing	X	/IWFND/SUP_REQ_DIR	Direction of the request
IV_OPERATION	importing	X	CHAR10	Character field length = 10
RO_LOGGER	returning		/IWFND/CL_LOGGER	Logger instance

Retrieving the Logger Instance During the Request Process to Create Log Entries

Use the method `GET_LOGGER` of class `/IWFND/CL_LOGGER` to get the logger instance for which you have to create the log messages.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
RO_LOGGER	returning		/IWFND/CL_LOGGER	Logger instance

Initiating the Business Processing on the SAP Gateway Server

Use the method `INIT_BUSINESS_PROCESS` of class `/IWFND/CL_LOGGER` to initiate the processing scope in which only business messages are created.

Finalizing the Business Processing on the SAP Gateway Server

Use the method `FINISH_BUSINESS_PROCESS` of class `/IWFND/CL_LOGGER` to finalize the processing scope in which only business messages are created. After this method only technical messages are created.

Logging a Step's Initiation

Use the method `LOG_STEP_INIT` of class `/IWFND/CL_LOGGER` to create the initiation log message.

Note

The type of an initiation message is always `I` and so does not need to be specified here.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IV_MSG_ID	importing		SYMSGID	Message class
IV_MSG_NUMBER	importing		SYMSGNO	Message number
IV_MSG_V1	importing	X	SYMSGV	Message variable
IV_MSG_V2	importing	X	SYMSGV	Message variable
IV_MSG_V3	importing	X	SYMSGV	Message variable
IV_MSG_V4	importing	X	SYMSGV	Message variable
IV_SYSTEM_ALIAS	importing	X	/IWFND/DEFI_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	importing	X	/IWFND/S_COR_ID	ID of the object whose processing caused the errors
IV_AGENT	importing		/IWFND/SUP_IW_AGENT	IW agent
RV_MSG_HANDLE	returning		BALMSGHNDL	Application Log: Message handle

1.1.5.1.2 Completing a Logging Step

Use

Use the method `LOG_STEP_COMPLETION` of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER` to log the completion of any step. If a step completed with an exception, use the method `LOG_STEP_COMPLETION_EXCEPTION`.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IV_MSG_TYPE	importing		SYMSGTY	Message type
IV_MSG_ID	importing		SYMSGID	Message class
IV_MSG_NUMBER	importing		SYMSGNO	Message number
IV_MSG_V1	importing	X	SYMSGV	Message variable
IV_MSG_V2	importing	X	SYMSGV	Message variable
IV_MSG_V3	importing	X	SYMSGV	Message variable
IV_MSG_V4	importing	X	SYMSGV	Message variable
IV_SYSTEM_ALIAS	importing	X	/IWFND/DEFI_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	importing	X	/IWFND/S_COR_ID	ID of the object whose processing caused the errors
IV_AGENT	importing		/IWFND/SUP_IW_AGENT	IW agent
IV_PROBLEM_CLASS	importing	X	BAL_S_MSG-PROBCLASS	Application log: Message problem class

IV_MSG_HANDLE	importing	X	BALMSGHNDL	Application Log: Message handle
IV_LOG_TO_CCMS	importing	X	/IWFND/SUP_LOG_CCMS	Flag for logging the message to CCMS

1.1.5.1.3 Logging an Exception

Use

Use the method `LOG_STEP_COMPLETION_EXCEPTION` of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER` to log an exception and to log the completion of the step which ended with this exception. The message type is not required as it is always an error.

This method creates the following log entries:

- Completed step
- The exception itself and corresponding description

The two entries are correlated, so the administrator can ascertain which exception log entry belongs to which step completion message.

Note

With regard to **applications**, don't forget to log each exception.

With regard to the **framework**, use this method to log the exceptions you have created or exceptions originating from an external system (backend, SAP Gateway).

Use the method `LOG_STEP_COMPLETION` for exceptions received from another SAP Gateway component and/or application. In this case, you only need to log the fact that the step ended with an exception.

- External exceptions:
For exceptions **not** of type `/IWFND/CX_BASE`, use the parameter `IO_EXCEPTION_EXTERN`. In this case, you must provide the parameters `IV_MSG_ID`, `IV_MSG_NUMBER` and `IV_MSG_V1-4`.
- SAP Gateway exceptions:
For SAP Gateway exceptions (of type `/IWFND/CX_BASE`), the message parameters can be maintained in the exception itself and do not need to be explicitly provided to the logger.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IO_EXCEPTION	importing	X	/IWFND/CX_BASE	IWFND base exception
IO_EXCEPTION_EXTERN	importing	X	CX_ROOT	Abstract superclass for all global exceptions
IV_MSG_ID	importing	X	SYMSGID	Message class
IV_MSG_NUMBER	importing	X	SYMSGNO	Message number
IV_MSG_V1	importing	X	SYMSGV	Message variable
IV_MSG_V2	importing	X	SYMSGV	Message variable
IV_MSG_V3	importing	X	SYMSGV	Message variable
IV_MSG_V4	importing	X	SYMSGV	Message variable
IV_SYSTEM_ALIAS	importing	X	/IWFND/DEFI_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	importing	X	/IWFND/S_COR_ID	ID of the object whose processing caused the errors
IV_AGENT	importing		/IWFND/SUP_IW_AGENT	SAP GW agent
IV_MSG_HANDLE	importing	X	BALMSGHNDL	Application Log: Message handle
IV_LOG_TO_CCMS	importing	X	/IWFND/SUP_LOG_CCMS	Flag for logging the message to CCMS

1.1.5.1.4 Logging BAPI Returns

Use

Use the method `LOG_BAPI_RETURN` of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER` to log messages returned in the BAPI return structure of an RFC.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IS_OBJECT_KEY	importing	X	/IWFND/S_COR_ID	ID of the object whose processing caused the errors
IT_BAPI_MESSAGES	importing		BAPIRET2_T	Table of BAPIRET2 messages
IV_AGENT	importing		/IWFND/SUP_IW_AGENT	SAP Gateway agent or component

1.1.5.1.5 Logging Simple Messages

Use

Use the method `LOG_MESSAGE` of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER` to log messages from the SAP Gateway message container. You can enter a T100 message, free text, or both.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IV_MSG_TYPE	importing		SYMSGTY	Message type
IV_MSG_ID	importing	X	SYMSGID	Message class
IV_MSG_NUMBER	importing	X	SYMSGNO	Message number
IV_MSG_TEXT	importing	X	/IWFND/SUP_MSG_LONGTEXT	Message text
IV_MSG_V1	importing	X	SYMSGV	Message variable
IV_MSG_V2	importing	X	SYMSGV	Message variable
IV_MSG_V3	importing	X	SYMSGV	Message variable
IV_MSG_V4	importing	X	SYMSGV	Message variable
IV_SYSTEM_ALIAS	importing	X	/IWFND/DEPI_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	importing	X	/IWFND/S_COR_ID	ID of the object whose processing caused the errors
IV_AGENT	importing		/IWFND/SUP_IW_AGENT	SAP GW agent
IV_PROBLEM_CLASS	importing	X	BAL_S_MSG-PROBCLASS	Application log: Message problem class
IV_MSG_HANDLE	importing	X	BALMSGHNDL	Application Log: Message handle
IV_LOG_TO_CCMS	importing	X	/IWFND/SUP_LOG_CCMS	Flag for logging the message to CCMS
IV_LOG_MSG_TEXT	importing	X	BOOLEAN	Boolean Variable (X=True, - =False, Space=Unknown)
RV_MSG_HANDLE	returning		BALMSGHNDL	Application Log: Message handle

1.1.5.1.6 Logging Simple Messages Through a Structure

Use

The message container uses the method `LOG_MESSAGE_STRUCTURE` of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER` to log its messages. The messages can be passed in a structure or as single values (message class, ID, type, variables).

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
IS_MESSAGE_STRUCTURE	importing		/IWFND/S_MECO_MESSAGE	Message type
IV_LOG_MSG_TEXT	importing	X	BOOLEAN	Boolean variable (X=True, - =False, Space=Unknown)
IV_MSG_HANDLE	importing	X	BALMSGHNDL	Application Log: Message handle
RV_MSG_HANDLE	returning		BALMSGHNDL	Application Log: Message handle

1.1.5.1.7 Logger Handling for Outbound Flows

Use

For outbound flows, the logger instance must be initiated and closed according to the start and end of a synchronous process.

A new logger instance must be created using the `INIT_LOGGER` method and closed using the `CLOSE_LOGGER` method (of class `/IWFND/CL_LOGGER` or `/IWBEP/CL_COS_LOGGER`) when the asynchronous process has finished. Between the logger initiation and closing, you can log messages as described in the previous sections.

CLOSE_LOGGER

Use the method `CLOSE_LOGGER` to save the log entries created during the request process to the Applications Log database.

If the entire process was successful (that is, only information messages were created), only one message is saved to the database, containing the information that the request was successfully processed on the SAP Gateway framework.

1.1.5.1.8 Non-User and Request-Specific Logging CCMS

Use

Use the method `LOG_MONITORING_INFORMATION` (or set the flag `IV_LOG_TO_CCMS` in the methods `LOG_MESSAGE`, `LOG_STEP_COMPLETION`, or `LOG_STEP_COMPLETION_EXCEPTION`) to report any type of information required by an administrator, ensuring all SAP Gateway components are running. The methods are in class `/IWFND/CL_LOGGER` and `/IWBEP/CL_COS_LOGGER`.

The following conditions may help identify the relevant problems:

- Problem can influence all users or can endanger the functionality of the entire component and/or application.
- Failure cannot be solved by retrying, or the maximum number of tries has been reached.
- Failure might not be immediately seen by the user; this can be an indicator for a potential problem.
- Information provided with the log can help the administrator or the support person to find or to solve the problem. For this reason, failed background processes must also be logged.
- SAP system failures must be logged to provide a complete picture.

Some examples are as follows:

- Consumer and/or SAP system connectivity layer is not configured correctly.
- Generic mapping of the inbound data to canonical internal data structures or of the internal data structures to SAP system data structures is not correct.
- The Document Publisher status and the Business Notification Sender queues have to be monitored (out-of-the box with qRFC).
- Background process ended in a failure.
- Status of scheduled tasks in the SAP system.

The method `LOG_MONITORING_INFORMATION` can consume logs of any severity type. The administrator sees a traffic light in the CCMS system that corresponds to the severity of the log.

CCMS can be configured to notify the administrator through mail or SMS if a log item of certain severity occurs. Exceptions have to be logged only at the level in which they can be handled appropriately. It is not recommended to log them on each instance they pass, since this can distort the results displayed in the analysis tools (Solution Manager Diagnostics and others).

BAPI return codes have to be translated into the corresponding set of logging attributes.

Note

Do not pass the content of BO items in any of the parameters.

IW Agent

Use only the values specified in the table `/IWFND/V_CCMS`, as the context structure defined in CCMS relies on them. If you try to log a message to CCMS (by setting the flag `IV_LOG_TO_CCMS` or by calling the method below) by using an agent not specified in the table `/IWFND/V_CCMS`, the following error message is created in the application log: Message 'message class' 'message ID' of the agent 'agent' could not be logged in CCMS.

Parameter Name	Parameter Type	Optional	Data Type	Def. Value
<code>IV_MSG_TYPE</code>	importing		<code>SYMSGTY</code>	Message type
<code>IV_MSG_ID</code>	importing		<code>SYMSGID</code>	Message class
<code>IV_MSG_NUMBER</code>	importing		<code>SYMSGNO</code>	Message number
<code>IV_MSG_V1</code>	importing	X	<code>SYMSGV</code>	Message variable
<code>IV_MSG_V2</code>	importing	X	<code>SYMSGV</code>	Message variable
<code>IV_MSG_V3</code>	importing	X	<code>SYMSGV</code>	Message variable
<code>IV_MSG_V4</code>	importing	X	<code>SYMSGV</code>	Message variable
<code>IV_SYSTEM_ALIAS</code>	importing	X	<code>/IWFND/DEFI_SYSTEM_ALIAS</code>	System alias
<code>IS_OBJECT_KEY</code>	importing	X	<code>/IWFND/S_COR_ID</code>	ID of the object whose processing caused the errors
<code>IV_AGENT</code>	importing		<code>/IWFND/SUP_IW_AGENT</code>	IW agent should be taken from table <code>/IWFND/I_COCCOL</code>

1.1.5.1.9 Message Container

Use

The message container gathers all messages raised and retrieved by application agents during the processing of a request. It is defined by the interface `/IWFND/IF_MESSAGE_CONTAINER` or `/IWBEP/IF_MESSAGE_CONTAINER`, which supports adding messages in the following different formats:

- Standard T100 format
- `BAPIRETURN2` table
- SOA Web service Log Item
- Exception

Adding Messages

An application should add all messages to the message container.

Return Code

A return code is not part of the message container but is a parameter of the request data object.

All SAP Gateway inbound Web Services support a return code of data type `/IWFND/COR_RESULT_CODE` (NUMC length 2).

Value	Description
OK	Success
WA	Processed with warnings
FP	Not processed successfully - permanent

1.1.5.2 OData Channel APIs

Use

In this section you can find a list of the most prominent OData Channel classes and interfaces for your reference.

- [OData Channel Runtime APIs](#)
- [OData Channel Metadata APIs](#)
- [OData Channel Push APIs](#)

In addition, you can use the central [/IWFND/CL_MGW_ACTIVATION_API](#).

1.1.5.2.1 OData Channel Runtime APIs

Use

The central classes and interfaces of OData Channel are as follows:

- [/IWBEP/CL_COS_LOGGER](#)
- [/IWBEP/CL_MGW_ABS_DATA](#)
- [/IWBEP/IF_MESSAGE_CONTAINER](#)
- [/IWBEP/IF_MGW_APPL_SRV_RUNTIME](#)
- [/IWBEP/IF_MGW_CONV_SRV_RUNTIME](#)
- [/IWBEP/IF_MGW_ENTRY_PROVIDER](#)
- [/IWBEP/IF_MGW_ODATA_EXPAND](#)
- [/IWBEP/CX_MGW_BUSI_EXCEPTION](#)
- [/IWBEP/IF_MGW_REQ_FUNC_IMPORT](#)
- [/IWBEP/IF_MGW_REQ_FILTER](#)
- [/IWBEP/IF_MGW_REQ_ENTITY_U](#)
- [/IWBEP/IF_MGW_REQ_ENTITY_D](#)
- [/IWBEP/IF_MGW_REQ_ENTITY_C](#)
- [/IWBEP/IF_MGW_REQ_ENTITY_P](#)
- [/IWBEP/IF_MGW_REQ_ENTITY](#)
- [/IWBEP/IF_MGW_REQ_ENTITYSET](#)
- [/IWBEP/IF_MGW_EXPR_VISITOR](#)

1.1.5.2.1.1 /IWBEP/CL_COS_LOGGER

Use

This API can be used to log failure information to the corresponding tools in SAP. These are the application log and the Computing Center Management System (CCMS).

Instantiation

```
DATA lo_logger TYPE REF TO /iwbep/cl_cos_logger.
```

```
" get logger object from context (/IWBEP/IF_MGW_CONTEXT) attribute of class /IWBEP/CL_MGW_ABS_DATA
lo_logger = me->mo_context->get_logger( ).
```

Methods

Method GET_LOGGER

This method returns the logger instance.

Method LOG_BAPI_RETURN

This method logs information provided in form of a `BAPIRET_2` structure.

Parameter	Description
IS_OBJECT_KEY	ID of the object causing the error
IT_BAPI_MESSAGES	BAPIRET_2 messages
IV_AGENT	Component which raises the message
IV_IS_EXTERNAL	External message get logged as text messages

Method LOG_FREE_TEXT_MESSAGE

Parameter	Description
IV_MSG_TEXT	Text of the message
IS_MSG	Type and context of the message text
RV_MSG_HANDLE	Message handle

Method LOG_MESSAGE

Parameter	Description
IV_MSG_TYPE	Message type
IV_MSG_ID	Message class
IV_MSG_NUMBER	Message number
IV_MSG_TEXT	Text of the message
IV_MSG_V1	Message variable
IV_MSG_V2	Message variable
IV_MSG_V3	Message variable
IV_MSG_V4	Message variable
IV_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	ID of the object causing the errors
IV_AGENT	Component
IV_PROBLEM_CLASS	Message problem class of application log
IV_MSG_HANDLE	Handle of previous message
IV_LOG_TO_CCMS	If set message will be forwarded to CCMS
IV_LOG_MSG_TEXT	If set the text of the message will be logged
RV_MSG_HANDLE	Message handle

Method LOG_MONITORING_INFORMATION

This method adds a message to CCMS.

Parameter	Description
IV_MSG_TYPE	Message type
IV_MSG_ID	Message class
IV_MSG_NUMBER	Message number
IV_MSG_V1	Message variable
IV_MSG_V2	Message variable
IV_MSG_V3	Message variable
IV_MSG_V4	Message variable
IV_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	ID of the object causing the error
IV_AGENT	Component

Method LOG_SECURITY_MESSAGE

This message logs a security message (using a security agent/component).

Parameter	Description
IV_MSG_TEXT	Text of the message
IV_EXT_KEY	External key of security message
IV_MSG_HANDLE	Handle of previous message
RV_MSG_HANDLE	Message handle

Method LOG_STEP_COMPLETION

This method logs the successful completion of a processing step.

Parameter	Description
IV_MSG_TYPE	Message type
IV_MSG_ID	Message class
IV_MSG_NUMBER	Message number

IV_MSG_V1	Message variable
IV_MSG_V2	Message variable
IV_MSG_V3	Message variable
IV_MSG_V4	Message variable
IV_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	ID of the object causing the error
IV_AGENT	Component
IV_PROBLEM_CLASS	Message problem class of application log
IV_MSG_HANDLE	Handle of previous message
IV_LOG_TO_CCMS	If set the message will be forwarded to CCMS

Method LOG_STEP_COMPLETION_EXCEPTION

This method logs the unsuccessful completion of a processing step.

Parameter	Description
IO_EXCEPTION_EXTERN	Exception to be logged
IV_MSG_ID	Message class
IV_MSG_NUMBER	Message number
IV_MSG_V1	Message variable
IV_MSG_V2	Message variable
IV_MSG_V3	Message variable
IV_MSG_V4	Message variable
IV_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	ID of the object causing the error
IV_AGENT	Component
IV_MSG_HANDLE	Handle of previous message
IV_LOG_TO_CCMS	If set the message will be forwarded to CCMS

Method LOG_STEP_INIT

This method creates the initial log message. Call this method first in your processing step.

Parameter	Description
IV_MSG_ID	Message class
IV_MSG_NUMBER	Message number
IV_MSG_V1	Message variable
IV_MSG_V2	Message variable
IV_MSG_V3	Message variable
IV_MSG_V4	Message variable
IV_SYSTEM_ALIAS	System alias
IS_OBJECT_KEY	ID of the object causing the error
IV_AGENT	Component
RV_MSG_HANDLE	Message handle

Method CLOSE_LOGGER

This method saves the application log and closes the logger object.

Example

Class /IWBEP/CL_MGW_RT_ERROR_MSG demonstrates how to use the logger in the following methods:

- TEST_LOGGER
- TEST_LOGGER_PROCESSING
- TEST_LOGGER_PROCESSING_EXC

More Information

For more information on supportability, the CCMS and the application log see the [SAP NetWeaver Gateway Foundation Technical Operations Guide](#).

1.1.5.2.1.2 /IWBEP/CL_MGW_ABS_DATA

This is the abstract class which acts as base class for every data provider class (DPC).

Use

This is the abstract class which acts as base class for every data provider class (DPC).

This class shall be used as a superclass by any OData Channel application (if no push support is needed) if the adaptation is done in the system in which the IW_BEP component is installed.

1.1.5.2.1.3 /IWBEF/IF_MESSAGE_CONTAINER

Use

The interface IWBEF/IF_MESSAGE_CONTAINER is used to add error details to the inner error section of the OData error response. See also <http://www.odata.org/media/30002/OData%20Atom%20Format.html#themetadata:errorelement> .

The format for the inner error section of the OData error response is as follows:

```
<innererror>
  <transactionid>123</transactionid>
  ..<errordetails>
    <errordetail>
      <message>Street must not be empty</message>
      <propertyref>Address/Street</propertyref>
      <severity>error</severity>
    </errordetail>
    ...
  </errordetails>
</innererror>
```

Each entry of the message container will be mapped to an **error detail** entry of the inner error and will also be logged in the SAP Gateway framework.

HTTP Header SAP-Messages

If a request is successful, warnings and success messages can be returned to the consumer in the HTTP header SAP-Messages. All messages that are added to the message container with the flag IV_ADD_TO_RESPONSE_HEADER = ABAP_TRUE are added to the HTTP header SAP-Messages if the request does **not** end in an error.

Example of how to add a message to the header SAP-Messages:

```
lo_message_container->add_message(
  iv_msg_type      = /iwbef/cl_cos_logger=>warning
  iv_msg_id        = /iwbef/cx_tea_business=>team_id_out_of_range-msgid
  iv_msg_number    = /iwbef/cx_tea_business=>team_id_out_of_range-msgno
  iv_add_to_response_header = abap_true
).
```

For more information, see [Map Message Container to Message Protocol Format](#)

Instantiation

```
DATA lo_message_container TYPE REF TO /iwbef/if_message_container.

" get message container object
lo_message_container = me->/iwbef/if_mgw_core_srv_runtime~mo_context->get_message_container( ).
```

Constants

Constant	Description	Value Set
GCS_MESSAGE_TYPE	Message types	abort, error, warning, success, info, termination
GCS_LEADING_MSG_SEARCH_OPTION	Method to determine the leading message	none, first, last
CS_SEVERITY	Severity of message	info, warning, error

Methods

Method ADD_ERROR_DETAIL

The method parameter IS_ERROR_DETAIL is a structure containing fields for message text, severity, code, and property reference:

- Code is a (custom) message code that a client can translate into a client-specific message text.
- Severity is the severity of the message.
- Property reference is the XPATH expression that points to the corresponding property for this message.

Method ADD_MESSAGE_FROM_EXCEPTION

Parameter	Description	Default Value
IO_EXCEPTION	Use this parameter to put your exception into the container	
IV_ERROR_CATEGORY	Error category	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_ERROR_CATEGORY-PROCESSING
IV_IS_LEADING_MESSAGE	The leading message will be displayed in the message	ABAP_FALSE

	section of the OData error response	
IV_ENTITY_TYPE	Entity type	
IT_KEY_TAB	Entity key as key-value table	
IV_ADD_TO_RESPONSE_HEADER	Flag for adding the message to the response header. If this parameter is set to ABAP_TRUE then the messages are reset in the response header.	ABAP_FALSE

Method ADD_MESSAGE_FROM_BAPI

Parameter	Description	Default Value
IT_BAPI_MESSAGES	Provide a table of BAPIRET2 messages	
IV_ERROR_CATEGORY	Error category	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_ERROR_CATEGORY-PROCESSING
IV_DETERMINE_LEADING_MSG	A leading message can optionally be determined by choosing the first or last error message in the table	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_LEADING_MSG_SEARCH_OPTION-NONE
IV_ENTITY_TYPE	Entity type	
IT_KEY_TAB	Entity key as key-value table	
IV_ADD_TO_RESPONSE_HEADER	Flag for adding the message to the response header. If this parameter is set to ABAP_TRUE then the messages are reset in the response header.	ABAP_FALSE

Method ADD_MESSAGE

Parameter	Description	Default Value
IV_MSG_TYPE	Message type	
IV_MSG_ID	Message class	
IV_MSG_NUMBER	Message number	
IV_MSG_TEXT	Text of the message	
IV_MSG_V1	Message variable	
IV_MSG_V2	Message variable	
IV_MSG_V3	Message variable	
IV_MSG_V4	Message variable	
IV_ERROR_CATEGORY	Error category	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_ERROR_CATEGORY-PROCESSING
IV_IS_LEADING_MESSAGE	Set if the message shall be the new leading message of the container	
IV_ENTITY_TYPE	Entity type	
IT_KEY_TAB	Entity key as key-value table	
IV_ADD_TO_RESPONSE_HEADER	Flag for adding the message to the response header. If this parameter is set to ABAP_TRUE then the messages are reset in the response header.	ABAP_FALSE

Method ADD_MESSAGES_FROM_LOG

Parameter	Description	Default Value
IT_LOG_MESSAGES	Table of log messages	
IV_ERROR_CATEGORY	OBSOLETE	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_ERROR_CATEGORY-PROCESSING
IV_ENTITY_TYPE	Entity type	
IT_KEY_TAB	Entity key as key-value table	

Method ADD_MESSAGE_TEXT_ONLY

Table 1:

Parameter	Description	Default Value
IV_MSG_TYPE	Message type	
IV_MSG_TEXT	Text of the message	
IV_ERROR_CATEGORY	OBSOLETE	/IWBEF/IF_MESSAGE_CONTAINER=>GCS_ERROR_CATEGORY-PROCESSING
IV_IS_LEADING_MESSAGE	Specify whether the message shall be the new leading message	
IV_ENTITY_TYPE	Entity type	
IT_KEY_TAB	Entity key as key-value table	

IV_ADD_TO_RESPONSE_HEADER	Flag for adding (or not adding) the message to the response header	ABAP_FALSE
---------------------------	--	------------

Method ADD_MESSAGES_FROM_CONTAINER

Note that this method is **obsolete**.

Parameter	Description	Default Value
IO_MESSAGE_CONTAINER	Reference to message container to be added	
IV_ADD_TO_RESPONSE_HEADER	Flag for adding the message to the response header. If this parameter is set to ABAP_TRUE then the messages are reset in the response header.	ABAP_FALSE

Method GET_MESSAGES

Parameter	Description	Default Value
IV_PROVIDE_TEXT	If set the text of the messages is inserted into the container structure	
RT_MESSAGES	Table of messages	

Method GET_LEADING_MESSAGE_TEXT

Parameter	Description	Default Value
RV_MESSAGE_TEXT	Text of the leading message	

Method GET_WORST_MESSAGE_TYPE

Parameter	Description	Default Value
RV_MESSAGE_TEXT	Worst message type inside the container	

Method GET_HAS_LEADING_MESSAGE

Parameter	Description	Default Value
RV_HAS_LEADING_MESSAGE	Is set if the container contains a leading message	

For more information, see [Error Response Control for Backend Data Provider](#) and [Map Message Container to Message Protocol Format](#).

1.1.5.2.1.4 /IWBEP/IF_MGW_APPL_SRV_RUNTIME

Use

This is the central interface implemented by every data provider class (DPC).

Methods

Method CREATE_ENTITY

This method is used for the create operation for an entity.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string.
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type string.
IO_DATA_PROVIDER	The interface to deserialize the payload into the application specific structure. Type /IWBEP/IF_MGW_ENTRY_PROVIDER
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEP/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEP/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEP/IF_MGW_REQ_ENTITY_C contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEP/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method CREATE_DEEP_ENTITY

This method is used for the create operation for an entity - deep insert.

This is then the operation to create an entity with deep data in an inlined format. Every deep insert request has to be handled by the implementation which has to decide whether it can fulfill the current deep insert request or not by a given expand expression.

The method signature is similar to the `CREATE_ENTITY` method except that it has an additional parameter `IO_EXPAND`. The SAP NetWeaver Gateway framework resolves the inlined data and translates it to an expand expression which is passed to the method via `IO_EXPAND`. It can be used to validate whether the current request including the inlined data matches a given expand expression or to retrieve purely the expand string. If the expand expression matches the data can be accessed via `IO_DATA_PROVIDER` similar to `CREATE_ENTITY`. The `ES_DATA` parameter expects a nested structure which contains the components for the inlined data. The structure components need to be named like the ABAP names of the navigation properties defined in the metadata.

The final response which is sent back to the consumer contains the newly created entry including the deep data if it is part of the return structure. This behavior is supported from SP05 onwards. If the return structure just contains the data in a flat form without the nested data the behavior does not change and the response does not contain any inlined data.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string.
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type string.
IO_DATA_PROVIDER	The interface to deserialize the payload into the application specific structure. Type /IWBEF/IF_MGW_ENTRY_PROVIDER
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH
IO_EXPAND	Represents the resolved expand expression for the inlines of the entry which is to be created.
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_C contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_DEEP_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method DELETE_ENTITY

This method can be used for a delete operation for an entity.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type string
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_D contains all request information and represents the requested entities in a technical format meaning with their technical naming.
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH

Method EXECUTE_ACTION

This is the method for the execution of an action with no direct relation to an entity type. It has service semantics.

Parameter	Description
IV_ACTION_NAME	Name of the action. Type string
IT_PARAMETER	Parameter. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_FUNC_IMPORT contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method GET_ENTITY

This is the method for the read operation for an entity/entry.

Parameter	Description
-----------	-------------

IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. F Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type /IWBEP/T_MGW_NAME_VALUE_PAIR
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEP/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEP/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEP/IF_MGW_REQ_ENTITY contains all request information and represents the requested entities in a technical format meaning with their technical naming
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEP/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.
ES_RESPONSE_CONTEXT	This parameter of type /IWBEP/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_ENTITY_CNTXT is used to specify for example a last-modified time stamp or a max-age (for cache handling).

Cache control on the client can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` and export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT`. If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, then SAP Gateway will generate the HTTP response header cache-control with a max-age directive, for example, cache-control: max-age=3600.

1 Note

If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, you must also set the export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` to '-' (meaning 'can be cached').

On the other hand, if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is set to 'X', then SAP Gateway will generate the following HTTP response header: cache-control: no-store, no-cache. The same is the case if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is left as undefined (that is, set to ''). An HTTP 304 response (not modified) can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` and export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED`.

The export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` should be filled with a time stamp that indicates when the last change of the related business data occurred. If the `ES_RESPONSE_CONTEXT-LAST_MODIFIED` is filled with a time stamp, then SAP Gateway will generate the HTTP response header last-modified with the related date, for example, last-modified: Tue, 27 Nov 2012 09:41:40 GMT. If the export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` contains a time stamp that is older than the HTTP request header if-modified-since then an HTTP 304 response ('not modified') will be sent by SAP Gateway (and in this case an HTTP response header last-modified will not be set). The export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED` may be set to 'X' only in case `LAST_MODIFIED` of the application data is older than the time stamp in HTTP request header if-modified-since. In such cases, the related application data is up-to-date and does not need to be sent at all. SAP Gateway will then generate an HTTP 304 response ('not modified').

Method GET_STREAM

This method can be used for the read operation for a media resource (binary) of a media link entry.

This is then used for the read operation for a media resource of a media link entry which retrieves a binary file which is associated to an entry. The request parameters are similar to a `GET_ENTITY` request. A media resource of a media link entry is requested via the URI in the `<atom:content>` `src` attribute. The URI can either be a link to an external resource or to an internal resource which is handled by the application via `GET_STREAM`.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type /IWBEP/T_MGW_NAME_VALUE_PAIR
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEP/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEP/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEP/IF_MGW_REQ_ENTITY contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_STREAM	The returned reference to the media resource structure which contains the binary data and the MIME type of a media resource. Use /IWBEP/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.
ES_RESPONSE_CONTEXT	This parameter of type /IWBEP/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_ENTITY_CNTXT is used to specify for example a last-modified time stamp or a max-age (for cache handling).

Cache control on the client can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` and export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT`. If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, then SAP Gateway will generate the HTTP response header cache-control with a max-age directive, for example, cache-control: max-age=3600.

1 Note

If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, you must also set the export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` to `'-'` (meaning 'can be cached').

On the other hand, if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is set to `'X'`, then SAP Gateway will generate the following HTTP response header: `cache-control: no-store, no-cache`. The same is the case if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is left as undefined (that is, set to `'`). An HTTP 304 response (not modified) can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` and export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED`.

The export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` should be filled with a time stamp that indicates when the last change of the related business data occurred. If the `ES_RESPONSE_CONTEXT-LAST_MODIFIED` is filled with a time stamp, then SAP Gateway will generate the HTTP response header `last-modified` with the related date, for example, `last-modified: Tue, 27 Nov 2012 09:41:40 GMT`. If the export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` contains a time stamp that is older than the HTTP request header `if-modified-since` then an HTTP 304 response ('not modified') will be sent by SAP Gateway (and in this case an HTTP response header `last-modified` will not be set). The export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED` may be set to `'X'` only in case `LAST_MODIFIED` of the application data is older than the time stamp in HTTP request header `if-modified-since`. In such cases, the related application data is up-to-date and does not need to be sent at all. SAP Gateway will then generate an HTTP 304 response ('not modified').

Method GET_EXPANDED_ENTITY

This method can be used for a read operation for an expanded entity / entry.

The read operation is similar to `GET_ENTITY` which has an additional expand reference to retrieve data in an inlined/deep format.

1 Note

Note that the SAP NetWeaver Gateway framework can completely handle expand requests in a generic way. In case of performance-critical scenarios or if you experience a bad response time it is highly recommended to handle complex full/subsets of the expand request by specific application implementations. Therefore it is required to redefine `GET_EXPANDED_ENTITY`. If you redefine `GET_EXPANDED_ENTITY` and the given expand expression cannot be handled by the application logic it is required to delegate to the super method or to the implementation of the super class.

The expand query option is passed via the object reference `IO_EXPAND`. It can be used to validate whether the current expand expression matches a given expand expression. If the expand expression matches the data reference, then `ER_ENTITY` can directly be filled by the implementation with data in a deep format. All navigation properties of an entity type including its sub paths are valid expand expressions to check and to handle by the application. All expand paths (separated by comma) which are handled by the implementation have to be passed back to the framework via the parameter `ET_EXPANDED_TECH_CLAUSES`.

Parameter	Description
<code>IV_ENTITY_NAME</code>	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
<code>IV_SOURCE_NAME</code>	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type <code>/IWBEF/T_MGW_NAME_VALUE_PAIR</code>
<code>IT_KEY_TAB</code>	Represents the keys or <code>NavPropCollection</code> of the first segment. Type <code>/IWBEF/T_MGW_NAME_VALUE_PAIR</code>
<code>IT_NAVIGATION_PATH</code>	The whole navigation path. Type <code>/IWBEF/T_MGW_NAVIGATION_PATH</code>
<code>IO_TECH_REQUEST_CONTEXT</code>	This parameter of type <code>/IWBEF/IF_MGW_REQ_ENTITY</code> contains all request information and represents the requested entities in a technical format meaning with their technical naming.
<code>IO_EXPAND</code>	Represents the expand expression for the navigation properties of the nested entry/feed which need to be inlined.
<code>ER_ENTITY</code>	The returned reference to the application specific structure which contains the data. In comparison to the <code>GET_ENTITY</code> method the <code>ER_ENTITY</code> reference is already pre-filled with a data reference of a deep structure component of the final expanded result structure. The <code>er_entity</code> can then be filled by the application specific implementation. If the given data reference is not used you can use your own reference and finally use convenience method <code>/IWBEF/CL_MGW_ABS_DATA-COPY_DATA_TO_REF</code> .
<code>ES_RESPONSE_CONTEXT</code>	This parameter of type <code>/IWBEF/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_ENTITY_CNTXT</code> is used to specify for example a last-modified time stamp or a max-age (for cache handling).
<code>ET_EXPANDED_CLAUSES</code>	This parameter is obsolete, use <code>ET_EXPANDED_TECH_CLAUSES</code> instead.
<code>ET_EXPANDED_TECH_CLAUSES</code>	This parameter is filled by the application and contains all expanded paths which have been handled by the application itself. This list of expand clauses is based on the technical (internal) navigation property names, not the external names

Cache control on the client can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` and export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT`. If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, then SAP Gateway will generate the HTTP response header `cache-control` with a max-age directive, for example, `cache-control: max-age=3600`.

1 Note

If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, you must also set the export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` to `'-'` (meaning 'can be cached').

On the other hand, if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is set to `'X'`, then SAP Gateway will generate the following HTTP response header: `cache-control: no-store, no-cache`. The same is the case if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is left as undefined (that is, set to `'`). An HTTP 304 response (not modified) can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` and export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED`.

The export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` should be filled with a time stamp that indicates when the last change of the related business data occurred. If the `ES_RESPONSE_CONTEXT-LAST_MODIFIED` is filled with a time stamp, then SAP Gateway will generate the HTTP response header `last-modified` with the related date, for example, `last-modified: Tue, 27 Nov 2012 09:41:40 GMT`. If the export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` contains a time stamp that is older than the HTTP request header `if-modified-since` then an HTTP 304 response ('not modified') will be sent by SAP Gateway (and in this case an HTTP response header `last-modified` will not be set). The export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED` may be set to `'X'` only in case `LAST_MODIFIED` of the application data is older than the time stamp in HTTP request header `if-modified-since`. In such cases, the related application data is up-to-date and does not need to be sent at all. SAP Gateway will then generate an HTTP 304 response ('not modified').

Method GET_ENTITYSET

This method can be used for a read/query operation for an expanded set of entities / feeds. This method for a (query) operation returns a list of entities, aka a feed, and so allows to do filtering, sorting, free text search, etc.

Server side paging can be achieved by setting a skiptoken. The data provider can decide to limit the amount of data which is sent back to the client in order to ensure performance. The skiptoken is defined by the application to indicate the next bulk of data and is passed to the gateway runtime to render the next link in the OData feed. The client then calls the next link including the skiptoken to retrieve the next set of data. The skiptoken from the next link can be accessed by the data provider via the technical request context. (Custom) query options are copied to the next link except the skiptoken itself, skip and top. The top parameter for client side paging is calculated by the gateway runtime with the following logic:

- \$top in request minus number of entities in response

Delta handling can be achieved by setting a delta token. The data provider can pass back a delta token, for example, a timestamp after reading the feed data in order to indicate that it supports a delta mode. After issuing a delta token for the first time all delta requests are dispatched to `GET_ENTITYSET_DELTA`. The delta token is then injected as a delta link at the end of the feed. If the client wants to get only the changed data it requests the delta via the delta link. A delta request is then handled by `GET_ENTITYSET_DELTA`. The default implementation of `GET_ENTITYSET_DELTA` forwards the request to `GET_ENTITYSET` (compatibility reasons). The following rules apply to the delta token:

- In case of service-side paging delta-link appears only in the last page - no next link anymore
- The delta-set is retrieved by requesting the delta link
- In case of server-side paging in combination with delta token the server can pass back a skiptoken on the first page of the delta
- The delta token is then passed in the next link and indicates that it is currently a kind of delta processing
- A response never contains a delta link and a next link
- A data provider cannot pass back a deltatoken and a skiptoken at the same time
- A next link can contain a deltatoken in analogy to \$filter or custom query options in case of server-side paging

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_FILTER_SELECT_OPTIONS	The filter parameters as an ABAP select options table for the filter parameter. Type /IWBEF/T_MGW_SELECT_OPTION
IT_ORDER	The details with regards to sorting as query options. Type /IWBEF/T_MGW_SORTING_ORDER
IS_PAGING	Paging information with the details of skip and top. Type /IWBEF/S_MGW_PAGING
IV_SEARCH_STRING	If search is used then it contains the search string. Type string
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. . Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITYSET contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.
ER_ENTITY_SET	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as

		convenience method.
ES_RESPONSE_CONTEXT		<p>This parameter of type <code>/IWBEF/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT</code> is used to specify:</p> <ul style="list-style-type: none"> • <code>inlinecount</code> (if requested via <code>has inlinecount</code>) • <code>count</code> (number of entries instead of the list if requested via <code>count</code>) • <code>skiptoken</code> (for server side paging) • <code>deltatoken</code> (for delta handling) • <code>expand_skiptokens</code> (for server side paging together with <code>expand</code>) • <code>last_modified</code> (for conditional reads) • <code>max_age</code> (for client cache control) • <code>is_not_modified</code> (for conditional reads) (See constants <code>/IWBEF/IF_MGW_APPL_TYPES=>GCS_MODIFICATION_STATUS</code>) • <code>do_not_cache_on_client</code> (for client cache control) (<code>/IWBEF/IF_MGW_APPL_TYPES=>GCS_CACHE_ON_CLIENT</code>) • <code>do_cache_and_page_on_hub</code> (to trigger caching on the hub in case softstate is enabled)

Cache control on the client can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` and export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT`. If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, then SAP Gateway will generate the HTTP response header `cache-control` with a max-age directive, for example, `cache-control: max-age=3600`.

Note

If export parameter `ES_RESPONSE_CONTEXT-MAX_AGE` is set to a value, you must also set the export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` to `'-'` (meaning 'can be cached').

On the other hand, if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is set to `'X'`, then SAP Gateway will generate the following HTTP response header: `cache-control: no-store, no-cache`. The same is the case if export parameter `ES_RESPONSE_CONTEXT-DO_NOT_CACHE_ON_CLIENT` is left as undefined (that is, set to `'`). An HTTP 304 response (not modified) can be achieved by setting export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` and export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED`. The export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` should be filled with a time stamp that indicates when the last change of the related business data occurred. If the `ES_RESPONSE_CONTEXT-LAST_MODIFIED` is filled with a time stamp, then SAP Gateway will generate the HTTP response header `last-modified` with the related date, for example, `last-modified: Tue, 27 Nov 2012 09:41:40 GMT`. If the export parameter `ES_RESPONSE_CONTEXT-LAST_MODIFIED` contains a time stamp that is older than the HTTP request header `if-modified-since` then an HTTP 304 response ('not modified') will be sent by SAP Gateway (and in this case an HTTP response header `last-modified` will not be set). The export parameter `ES_RESPONSE_CONTEXT-IS_NOT_MODIFIED` may be set to `'X'` only in case `LAST_MODIFIED` of the application data is older than the time stamp in HTTP request header `if-modified-since`. In such cases, the related application data is up-to-date and does not need to be sent at all. SAP Gateway will then generate an HTTP 304 response ('not modified').

Method GET_ENTITYSET_DELTA

This method provides the read/query delta operation for a set of entities / feed including deleted entries. This method can be used to handle requests which contain a delta token. See also `GET_ENTITYSET` for server side paging and the initial delta handling. Based on the incoming delta token the server calculates the delta which needs to be sent back to the client. The modified data is passed back to the SAP Gateway runtime via `ER_ENTITYSET`. The deleted entries are passed back via the `ER_DELETED_ENTITYSET` data reference.

Note

Tombstone support is only available for Atom (XML) but not for JSON. Expand is not supported in combination with tombstones.

The Atom "deleted-entry" element (also known as Tombstone) is specified in proposed standard RFC 6721. It is used to explicitly identify removed Atom entries in Atom feeds.

```
deletedEntry =
  element at:deleted-entry {
    atomCommonAttributes,
    attribute ref { atomUri },
    attribute when { atomDateConstruct },
    ( element at:by { atomPersonConstruct }?
      & element at:comment { atomTextConstruct }?
      & element atom:link { atomLink }*
      & element atom:source { atomSource }?
      & anyElement* )
```

The attribute `ref` is set to the same value as the `atom:id` element. Attribute `when` is set to the value of the property that is mapped to `SyndicationUpdated` or - if such a mapping does not exist - to the current timestamp (same as for `atom:updated`).

`at:deleted-entry` elements appear en bloc after all `atom:entry` elements in the Atom feed.

Table 1:

Parameter	Description
IO_TECH_REQUEST_CONTEXT	This parameter of type <code>/IWBEF/IF_MGW_REQ_ENTITYSET</code> contains all request information and represents the requested entities in a technical format, that is, with their technical naming.
ER_ENTITYSET	The returned reference to the application-specific structure which contains the data. Use <code>/IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF</code> as convenience method.

ER_DELETED_ENTITYSET	The returned reference to the application-specific structure which contains the data of the deleted entries (tombstones). Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method. This table should be of the same type as the table for parameter ER_ENTITYSET.
ES_RESPONSE_CONTEXT	This parameter of type /IWBEF/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT is used to specify a skiptoken for example (for server-side paging), the inline count (if requested via has inline count), the count or a delta token.

Method GET_EXPANDED_ENTITYSET

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_FILTER_SELECT_OPTIONS	The filter parameters as an ABAP select options table for the filter parameter. Type /IWBEF/T_MGW_SELECT_OPTION
IT_ORDER	The details with regards to sorting as query options. Type /IWBEF/T_MGW_SORTING_ORDER
IS_PAGING	Paging information with the details of skip and top. Type /IWBEF/S_MGW_PAGING
IV_SEARCH_STRING	If search is used then it contains the search string. Type string
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITYSET contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITYSET	The returned reference to the application specific table which contains the data. In difference to the GET_ENTITYSET method the ER_ENTITYSET reference is already pre-filled with a data reference of a deep structure component of the final expanded result feed. The ER_ENTITYSET can then be filled by the application specific implementation. If the given data reference is not used the reference needs to be filled, for example by using convenience method /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF. The data is then copied into the nested result.
ES_RESPONSE_CONTEXT	This parameter of type /IWBEF/IF_MGW_APPL_SRV_RUNTIME=>TY_S_MGW_RESPONSE_CONTEXT is used to specify the request attribute. For example, a skiptoken (for server side paging), the inline count (if requested via has inline count) or a delta token.
ET_EXPANDED_CLAUSES	This parameter is obsolete, use ET_EXPANDED_TECH_CLAUSES instead.
ET_EXPANDED_TECH_CLAUSES	This parameter is filled by the application and contains all expanded paths which have been handled by the application itself. This list of expand clauses is based on the technical (internal) navigation property names, not the external names

The (query) operation similar to GET_ENTITYSET which has an additional expand reference to retrieve data in an inlined/deep format. Note that the SAP Gateway framework can completely handle expand requests in a generic way. In case of performance-critical scenarios or if you experience a bad response time it is highly recommended to handle complex full/subsets of the expand request by specific implementations. Therefore if you require to redefine GET_EXPANDED_ENTITYSET. If you redefine GET_EXPANDED_ENTITYSET and the given expand expression cannot be handled by the application logic it is required to delegate to the super method or to the implementation of the super class. The expand query option is passed via the object reference IO_EXPAND. It can be used to validate whether the current expand expression matches a given expand expression. If the expand expression matches the data reference, then ER_ENTITY can directly be filled by the implementation with data in a deep format. All navigation properties of an entity type including its sub paths are valid expand expressions to check and to handle by the application. All expand paths (separated by comma) which are handled by the implementation have to be passed back to the framework via the parameter ET_EXPANDED_TECH_CLAUSES.

Method UPDATE_ENTITY

This method updates an entity.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IO_DATA_PROVIDER	The interface to deserialize the payload into the application specific structure.

	Type /IWBEF/IF_MGW_ENTRY_PROVIDER
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_U contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method PATCH_ENTITY

A patch request is a partial update of an entity. All provided components in the request are patched.

The default implementation of `PATCH_ENTITY` performs a read before update without locking the corresponding business object. If this is required the default implementation can be used but the `PATCH_ENTITY` method needs to be overwritten, and after an enqueue the default implementation can be called. After that the business object has to be de-queued. All unpatched simple properties of the entity type are copied during the read before update. Complex properties are treated in the same way and are merged with the original values if not given in the request.

A sample implementation for the lock semantics is provided in the last section of the default implementation

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation.
IO_DATA_PROVIDER	The interface to deserialize the payload into the application specific structure. Type /IWBEF/IF_MGW_ENTRY_PROVIDER
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_U contains all request information and represents the requested entities in a technical format meaning with their technical naming. In addition to the information provided by the <code>UPDATE_ENTITY</code> it contains a nested components table with the patched properties in the <code>PATCH</code> request.
ER_ENTITY	The returned reference to the application specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method UPDATE_STREAM


This method updates a stream.

Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_ENTITY_SET_NAME	
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation.
IS_MEDIA_RESOURCE	Content type and content value. For example, Content type = image/jpg and content value in binary format. Type <code>TY_S_MEDIA_RESOURCE</code>
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH (table of navigation paths).
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_U contains all request information and represents the requested entities in a technical format meaning with their technical naming.

Method CREATE_STREAM

This method creates a stream (media resource) and the corresponding entity (media link entry). Additional information can be provided in the Slug header. The choreography of a media link entry creation is to do a HTTP Post first which contains the binary data only. Note that it is not allowed to post the entry itself first and later put the associated media resource. Based on the Slug header you can pass additional information to the server which can be used for storing the binary (media resource) and to create the skeleton of the entity (media link entry) creation, for example. The newly created entity (media link entry) is sent back to the consumer in the HTTP response which can be updated with the proper data. The updated entry is sent to the SAP NetWeaver Gateway system in an HTTP Put to complete the process of media resource/media link entry creation.

Parameter	Description
-----------	-------------

IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_ENTITY_SET_NAME	
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation.
IS_MEDIA_RESOURCE	Content type and content value. For example, Content type = image/jpg and content value in binary format. Type TY_S_MEDIA_RESOURCE
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH (table of navigation paths).
IV_SLUG	Slug is a special HTTP header, see http://bitworking.org/projects/atom/rfc5023#rfc.section.9.7  .
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_C contains all request information and represents the requested entities in a technical format meaning with their technical naming.
ER_ENTITY	The returned reference to the application-specific structure which contains the data. Use /IWBEF/CL_MGW_ABS_DATA~COPY_DATA_TO_REF as convenience method.

Method DELETE_STREAM

This method deletes a stream. The corresponding entity is not touched.


Parameter	Description
IV_ENTITY_NAME	Name of the entity type which is requested. If there is a navigation it means that it represents the end/target of the navigation path. Type string
IV_ENTITY_SET_NAME	
IV_SOURCE_NAME	Name of the entity type which is at the beginning of the navigation path, if there is a navigation.
IT_KEY_TAB	Represents the keys or NavPropCollection of the first segment. Type /IWBEF/T_MGW_NAME_VALUE_PAIR
IT_NAVIGATION_PATH	The whole navigation path. Type /IWBEF/T_MGW_NAVIGATION_PATH (table of navigation paths).
IO_TECH_REQUEST_CONTEXT	This parameter of type /IWBEF/IF_MGW_REQ_ENTITY_D contains all request information and represents the requested entities in a technical format meaning with their technical naming.

Method: CHANGESSET_BEGIN

Table 2:

Parameter	Description
IT_OPERATION_INFO	Table containing the following information of all operations in the current changeset (/IWBEF/T_MGW_OPERATION_INFO). <ul style="list-style-type: none"> ENTITY_NAME as STRING - Name of the entity type which is requested. ENTITY_NAME as STRING - Name of the entity type which is requested. ACTION_NAME as STRING - Name of the action if execution of an action with no direct relation to an entity type.
CV_DEFER_MODE	Indicates that the data provider will process the current changeset in defer mode (later at calling of method CHANGESSET_PROCESS).

All operations within a changeset must be treated as a Logical Unit of Work. This means all or nothing. Therefore, a provider must NOT issue COMMIT WORK or ROLLBACK WORK during a changeset processing. Otherwise, the framework will abandon the changeset processing. If the changeset contains only one operation, the check of commit or rollback is deactivated. At the beginning of a changeset processing, the provider will be called with this API. It can check the list of all involved entity types and actions containing in this changeset and accept the changeset handling or reject by raising a technical exception with exception code CHANGESSET_NOT_SUPPORTED. The method has a default implementation which allows only one operation per changeset in order to guarantee the transactional consistency. If more than one operation is required in a changeset the implementation has to be overwritten by the application. The application then needs to ensure the transactional consistency e.g. not to have any commit or rollback in the chain of requests in a changeset.

Besides that the provider can start to collect the content of a changeset in the modifying method calls e.g. update_entity and finalize it in the changeset_end method call. From SAP Gateway 2.0 SP 07 onwards, the check of COMMIT WORK and ROLLBACK WORK is only active from CHANGESSET_BEGIN until before CHANGESSET_END is called. This means a provider can issue COMMIT WORK or ROLLBACK WORK from within the processing of CHANGESSET_END. This is useful for providers which reuse other components including COMMIT WORK handling. It is also necessary for providers which have to do some post-processing works after COMMIT WORK is called. This check modification is also available for SAP Gateway 2.0 SP 06 via SAP Note 1824626 .

1 Note

Do not issue COMMIT WORK or ROLLBACK WORK during changeset processing if the changeset contains more than one operation. COMMIT WORK or ROLLBACK WORK is only possible at CHANGESSET_END. The framework checks COMMIT WORK calls during changeset processing and terminates batch processing with a short dump if errors arise. Consequently, you should also not call any coding that would trigger the same behavior.

From SAP Gateway 2.0 SP 09 onwards, a data provider can check the entire changeset operations in `IT_OPERATION_INFO` and return `CV_DEFER_MODE = 'X'` to indicate that it does not process the entire changeset operations immediately but only saves these operations until the method `CHANGESET_PROCESS` is called. A data provider can dynamically decide to process the current changeset in "defer mode" based on the entire operation info described in table `IT_OPERATION_INFO`.

In defer mode, when `CREATE`, `UPDATE`, `DELETE` or `EXECUTE_ACTION` is called the data provider has only to save all changeset operations in its internal tables without returning any data or ETag or headers to the framework.

Method `CHANGESET_PROCESS`

The data provider has to process all entire changeset operations and return the operation results including ETags, headers in the response table. Warning and info messages (SAP messages) for a specific operation must be set by using the message container instance stored in each operation entry of the request table. During the processing of `CHANGESET_PROCESS`, the data provider must not use the methods `SET_ETAG` or `SET_HEADERS` (technical exception with specific error text will be raised). It has to pass this information in the response table `CT_CHANGESET_RESPONSE`. In the system you can find sample coding in class `/IWBEP/CL_MGW_RT_SFLIGHT`, method `CHANGESET_BEGIN` and `CHANGESET_PROCESS`.

Table 3:

Parameter	Description
<code>IT_CHANGESET_REQUEST</code>	Request table containing all operation data of the current changeset (<code>/IWBEP/IF_MGW_APPL_TYPES=>TY_T_CHANGESET_REQUEST</code>).
<code>CT_CHANGESET_RESPONSE</code>	Response table containing the results of all changeset operations (<code>/IWBEP/IF_MGW_APPL_TYPES=>TY_T_CHANGESET_RESPONSE</code>).

Method `CHANGESET_END`

In case if the provider only updates all modifications of a changeset in internal tables, it can now update the database. A `COMMIT WORK` will be issued by the framework at the end of this API.

Method `CREATE_LINK`

This method can be implemented to add associations between two entities by using \$links.



Example

Example request: Post to the URL:

```
https://<server>:<port>/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/ServiceCollection('ZTEA_TEST_APPLICATION_0001')/$links/TagCollection
With the payload: <uri xmlns="http://schemas.microsoft.com/ado/2007/08/dataservices">
https://<server>:<port>/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/TagCollection('TEA') </uri> End of the example.
```

Method `DELETE_LINK`

This method can be implemented to delete \$links associations between two entities



Example

Example request: DELETE to the URL:

```
https://<server>:<port>/sap/opu/odata/IWFND/CATALOGSERVICE;v=2/ServiceCollection('ZTEA_TEST_APPLICATION_0001')/$links/TagCollection('TEA')
```

Method `GET_IS_CONDITIONAL_IMPLEMENTED`

This method can be implemented to indicate that conditional handling is implemented in the application. If the conditional handling is done for the given operation type and entity set name in the application the method must return `ABAP_TRUE`.

Table 4:

Parameter	Description
<code>IV_OPERATION_TYPE v</code>	Type <code>/IWBEP/MGW_OPERATION_TYPE</code> .
<code>IV_ENTITY_SET_NAME</code>	String
<code>RV_CONDITIONAL_ACTIVE</code>	<code>ABAP_BOOL</code>

<http://www.odata.org/documentation/odata-v3-documentation/odata-core/>

1.1.5.2.1.5 `/IWBEP/IF_MGW_CONV_SRV_RUNTIME`

Use

This interface offers several convenience methods provided by the framework.

Methods

Method `COPY_DATA_TO_REF`

This is a convenience method to get a data reference of a type any variable.

Parameter	Description
<code>IS_DATA</code>	Variable of type any which has to be either a structure or an internal table.
<code>CR_DATA</code>	Data reference of the imported variable.

Method `CHECK_IF_IS_NOT_MODIFIED`

This method can be used to fill `es_response_context-is_not_modified`. It checks if data was not modified based on:

- if-modified-since HTTP request header (time stamp)
- last-modified time stamp provided by the application

In case last-modified is older than if-modified-since then the data is up-to-date. In this case the returning parameter `rv_is_not_modified` is set to 'X' (/iwbeb/if_mgw_appl_types=>gcs_modification_status-is_not_modified) This method can be used to fill `es_response_context-is_not_modified`. In case `es_response_context-is_not_modified` is set to 'X' (/iwbeb/if_mgw_appl_types=>gcs_modification_status-is_not_modified) then an HTTP response 304 ,Not Modified' shall be sent.

Table 1:

Parameter	Description
IV_IF_MODIFIED_SINCE	Variable of type TZNTSTMPs, the if-modified-since time stamp
IV_LAST_MODIFIED	Variable of type TZNTSTMPs, the last-modified-since time stamp.
RV_IS_NOT_MODIFIED	Variable of type /IWBEF/MGW_IS_NOT_MODIFIED.

Method SET_HEADER

This method adds the header parameters to the HTTP response in a key/value approach.

Parameter	Description
IS_HEADER	Name/value pair as header parameter of the HTTP response. By setting this parameter you can, for example, influence the caching behavior.

Method SET_ETAG

This method sets the HTTP ETag in the HTTP response header.

Note

This method is only needed in the OData Compatibility Mode for SP 02 . The ETag needs to be set in the `READ_ENTITY` method with the value of the property which is marked as ETag. You do not have to set the ETag in the OData Standard Mode .

Parameter	Description
IV_VALUE	ETag value

Method GET_LOGGER

This method returns an instance of the OData Channel logger. It should be used by the DPC (data provider class) for logging purposes.

Parameter	Description
RO_LOGGER	Instance

Method GET_MESSAGE_CONTAINER

This method returns an instance of the OData Channel message container. It should be used by the DPC to pass messages back to the framework.

Parameter	Description
RO_MESSAGE_CONTAINER	Instance

Method GET_RESPONSE_UTIL

This method returns an instance of an utility that supports the creation of specific information in the response message.

Parameter	Description
RO_RESPONSE_UTIL	Instance

The response utility mainly provides two methods `GET_TARGET_IN_ENTITY` and `GET_TARGET_IN_ENTITY_SET` to construct syntactically correct target information for error messages or warnings (see method signatures of the message container). Input: the technical entity and/or entity set name, key information if required, and a path to a property in ABAP format. The methods return a target string that already has the required format or will be translated into the final format on the hub.

The target calculation method `GET_TARGET_IN_ENTITY_SET` at the response utility /IWBEF/CL_MGW_RESPONSE_UTIL has been extended to support specification of a message target relative to the service. In this case, the final message target starts with '/', for example, '/Managers('0001')'. A new optional importing parameter has been added. The information is encoded into the preliminary message target. The hub logic constructs the final target based on the new parameter, the entity set name, and all subsequent target segments.

Usage of method `GET_TARGET_FROM_INTERNAL_NAMES` should be replaced by `GET_TARGET_IN_ENTITY`.

Method GET_DP_FACADE

This method returns a facade which is reserved for future features supporting content development. Currently it does not contain any methods.

Parameter	Description
RO_DP_FACADE	To access special utility methods

Method INIT_DP_FOR_UNIT_TEST

This method is used to initialize a data provider instance for the usage in a unit test.

1. An instance of the new request context (/IWBEF/CL_MGW_REQUEST_UNITTST) is created.
2. The request values are assigned to the request context instance.
3. Member variables of the /IWBEF/CL_MGW_ABS_DATA are initialized and set:

- MO_CONTEXT
- MR_REQUEST_DETAILS
- MR_SERVICE_DOCUMENT_NAME
- MR_SERVICE_VERSION
- MR_SERVICE_NAMESPACE

4. Logger and message container instances are retrieved

Table 2:

Parameter	Description
IS_REQUEST_CONTEXT	Structure containing the new request context for unit tests, TYPE /IWBEP/CL_MGW_REQUEST_UNITTST=>TY_S_MGW_REQUEST_CONTEXT_UNIT
RO_REQUEST_CONTEXT	Request context object for further usage through the DPC. TYPE REF TO /IWBEP/CL_MGW_REQUEST_UNITTST

1.1.5.2.1.6 /IWBEP/IF_MGW_ENTRY_PROVIDER

Use

This interface can be used to access and deserialize the data of an update/create (PUT/POST) action from the consumer.

Methods

Method READ_ENTRY_DATA

This method provides the entry point for getting the data either of an entry which was sent via a HTTP POST (create) or via a HTTP PUT (update) request.

Parameter	Description
ES_DATA	Reference to a structure to which the data should be deserialized. In case of a simple create/update es_data is bound to a flat structure. In case of a deep insert a nested structure is expected.

1.1.5.2.1.7 /IWBEP/IF_MGW_ODATA_EXPAND

This interface can be used to validate whether an expand expression which can be handled by the application is applicable or not.

Moreover, this interface provides also some more methods for service providers to get the whole expand tree and also retrieve all necessary information to implement its own expand completely or just partly to optimize performance.

The following fields are provided:

- GCS_COMPARE_RESULT-MATCH_NO
- GCS_COMPARE_RESULT-MATCH_SUBSET
- GCS_COMPARE_RESULT-MATCH_FULL

Methods

Method COMPARE_TO

Note

This method is **obsolete**. Use method COMPARE_TO_TECH_NAMES instead.

This method compares a given string with the requested (sub)expand expression. Instead of technical names it is required to provide the external naming to compare the expand expression. This is not recommended.

Parameter	Description
IV_EXPAND	Expression which can be handled by the application
RV_RESULT	This has the value GCS_COMPARE_RESULT-MATCH_NO if the given expand string does not match. It has the value GCS_COMPARE_RESULT-MATCH_SUBSET if the given expand is a subset of the current expand tree. It has the has value GCS_COMPARE_RESULT-MATCH_FULL if the given expand string matches fully.

Method COMPARE_TO_TECH_NAMES

This method compares a given string with the requested (sub)expand expression.

Parameter	Description
IV_EXPAND	Expression which can be handled by the application
RV_RESULT	This has the value GCS_COMPARE_RESULT-MATCH_NO if the given expand string does not match. It has the value GCS_COMPARE_RESULT-MATCH_SUBSET if the given expand is a subset of the current expand tree. It has the has value GCS_COMPARE_RESULT-MATCH_EQUALS if the given expand string matches fully.

The expand handling of SAP Gateway allows to handle certain expand expressions by the data providers without applying the generic expand functionality. It is up to the data provider developer to check for certain expand expressions which can be handled in an efficient way on the application side. The `COMPARE_TO` method can be used to check for several expand expressions which shall be handled by the application logic.

In the `GET_EXPANDED_ENTITY` / `GET_EXPANDED_ENTITYSET` methods you can use the comparator to check whether the expand expression should be delegated to the framework or whether it can be handled by the application. It can lead to the following results:

- It returns `MATCH_SUBSET` if you call `COMPARE_TO_TECH_NAME(IV_EXPAND = 'PRODUCTS')` - which is the indicator that you can return a table of categories in which every category entry has a 'PRODUCTS' table component. The supplier relation is in this case further expanded generically by the framework. Exporting parameter `ET_EXPANDED_TECH_CLAUSES` of `GET_EXPANDED_ENTITY` / `GET_EXPANDED_ENTITYSET` needs then to contain one entry 'PRODUCTS' to tell the framework that this expression has already been handled by the application.
- It returns `MATCH_EQUALS` if you call `COMPARE_TO_TECH_NAME(IV_EXPAND = 'PRODUCTS/SUPPLIER')` - which is the indicator that you can return a table of categories in which every category entry has a 'PRODUCTS' table component and every products entry contains another structure component 'SUPPLIER'. Exporting parameter `ET_EXPANDED_TECH_CLAUSES` of `GET_EXPANDED_ENTITY` / `GET_EXPANDED_ENTITYSET` needs then to contain one entry 'PRODUCTS/SUPPLIER' to tell the framework that this expression has already been handled by the application. If the \$expand expression follows several navigation paths for the category which are handled by application the `ET_EXPAND_TECH_CLAUSES` need to contain an additional entry besides 'PRODUCTS/SUPPLIER' for each path.

Method `GET_TECH_ENTITY_SET`

This method returns the technical name of an entity set.

Parameter	Description
<code>RV_ENTITY_SET</code>	Type /IWBEF/MGW_TECH_NAME

Method `GET_TECH_ENTITY_TYPE`

This method returns the technical name of an entity type.

Parameter	Description
<code>RV_ENTITY_TYPE</code>	Type /IWBEF/MGW_TECH_NAME

Method `GET_MULTIPLICITY`

This method returns the cardinality 0..1, 1..1, 0..n or 1..n.

Parameter	Description
<code>RV_MULTIPLICITY</code>	Type /IWBEF/MGW_MED_ODATA_TYPES=>TY_E_MED_CARDINALITY

Method `GET_CHILDREN`

This method returns the child nodes of an entity.

Parameter	Description
<code>RT_CHILDREN</code>	Type /IWBEF/IF_MGW_ODATA_EXPAND=>TY_T_NODE_CHILDREN

Method `GET_SELECT_PROPERTIES`

This method is similar to `GET_SELECT_WITH_MANDTRY_KEYS` of interface `/IWBEF/IF_MGW_REQ_ENTITY` or `/IWBEF/IF_MGW_REQ_ENTITYSET` and returns the selected properties including the mandatory fields (keys, etag, mime type mapping) of the current entity and also of all included expands.

Parameter	Description
<code>RT_SELECT</code>	Type STRING_TABLE

Method `GET_SELECT_ENTITY_PROPERTIES`

This method is similar to `GET_SELECT_ENTITY_PROPERTIES` of interface `/IWBEF/IF_MGW_REQ_ENTITY` or `/IWBEF/IF_MGW_REQ_ENTITYSET` and contrary to `GET_SELECT_PROPERTIES`, it only returns the selected properties including the mandatory fields of the current entity.

Parameter	Description
<code>RT_SELECT</code>	Type STRING_TABLE

Method `IS_EXPAND_NEEDED`

This method is provided to prevent unneeded expand, in order to improve processing performance.

Example: Using the request URI "...Orders(OrderId='1')?\$expand=OrderDetails&\$select=OrderDate" an OData consumer causes an expand from the order entity with key OrderId '1' to OrderDetail, but it only selects the property OrderDate from the order entity. That means the expand to OrderDetail is actually not needed, and the provider application can ignore the expand requirement.

Note

This method only returns the correct value if the SAP Gateway system is also on at least support package 11. Otherwise, it always returns ABAP_TRUE.

Parameter	Description
RV_VALUE	Type ABAP_BOOL

1.1.5.2.1.8 /IWBEP/CX_MGW_BUSI_EXCEPTION

Use

The business exception CX_MGW_BUSI_EXCEPTION is provided by the OData Channel framework for exception handling.

The message text of the OData error response is the exception text in case of direct usage or the leading message of the message container in case of message container usage.

Instantiation

Message container usage:

```
RAISE EXCEPTION TYPE /iwbsp/cx_mgw_busi_exception
EXPORTING
  message_container = io_message_container.
```

Direct usage:

```
RAISE EXCEPTION TYPE /iwbsp/cx_mgw_busi_exception
EXPORTING
  textid                = /iwbsp/cx_mgw_busi_exception=>business_error_unlimited
  message_unlimited = 'Your message text'.
```

Attributes

Attribute	Data Type	Entity Type	Description
ENTITY_TYPE	STRING	RESOURCE_NOT_FOUND	Entity type causing/affected by the error
MESSAGE	BAPI_MSG	BUSINESS_ERROR	Message text
MESSAGE_UNLIMITED	STRING	BUSINESS_ERROR_UNLIMITED	Message text
HTTP_STATUS_CODE	/IWBEP/MGW_HTTP_STATUS_CODE		HTTP status code. For supported values see the constants in CS_HTTP_STATUS_CODES of exception class /IWBEP/CX_MGW_BUSI_EXCEPTION.
HTTP_HEADER_PARAMETERS	/IWBEP/T_MGW_NAME_VALUE_PAIR		HTTP header parameters

1.1.5.2.1.9 /IWBEP/IF_MGW_REQ_FUNC_IMPORT

Use

Interface to provide all request information relevant for function import. All artifacts are returned in a technical format meaning with their technical naming.

Methods

Method GET_FUNCTION_IMPORT_NAME

This method returns the names of the function import.

Parameters	Description
RV_ACTION_NAME	This parameter of type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME returns the names of the function import

GET_PARAMETERS

This method returns the table of function import parameters.

Parameters	Description
ET_PARAMETERS	This parameter of type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>PARAMETER_VALUES_T returns the table of function import parameters.

GET_FUNCTION_RETURN_TYPE

This method returns the entity type of the function import.

Parameters	Description
RV_RETURN_TYPE	String

GET_CONVERTED_PARAMETERS

If you want to use this method, the model provider class (MPC) must call method `BIND_INPUT_STRUCTURE()` of the current action (function import) after having issued one or more `CREATE_INPUT_PARAMETER()` to define the action parameters.

Parameter `ES_PARAMETER_VALUES` must be a structure which contains all defined parameters including the reference fields for currency or quantity if necessary (that is, parameter value is an amount).

If convertible field is used as parameter and the reference field for currency or quantity is needed, the reference field must also be defined as parameter. Otherwise, the returned parameter value cannot be converted and remains unchanged.

If a parameter field is missing in structure `ES_PARAMETER_VALUES` the parameter value will not be provided to the provider application.

If one or more parameters are defined in the model provider class as not convertible the returned parameter values are not converted. Therefore, this method can be used in general instead of `GET_PARAMETERS`. Do not forget to extend your MPC as explained above.

Parameters	Description
ES_PARAMETER_VALUES	Type DATA

1.1.5.2.1.10 /IWBEP/IF_MGW_REQ_FILTER

Use

This is the filter object representing \$filter parameter.

Methods

Method GET_FILTER_SELECT_OPTIONS

This method is used to return the filter expression parsed to select options table

Parameter	Description
RT_FILTER_SELECT_OPTIONS	This parameter of type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>EDM_SELECT_OPTION_T returns the filter expression parsed to select options table

Method GET_FILTER_STRING

This method is used to return the filter string in format similar to SQL `WHERE` clause labels parameters.

Parameter	Description
RV_FILTER_STRING	String

1.1.5.2.1.11 /IWBEP/IF_MGW_REQ_ENTITY_C

Use

This Interface is developed to provide all request information relevant for entity `CREATE` operation. All artifacts are returned in a technical format meaning with their technical naming.

Methods

Method GET_ENTITY_TYPE_NAME

This method is used to return the technical name of the target entity type

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_ENTITY_SET_NAME

This method is used to return the technical name of target the entity set.

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_SOURCE_ENTITY_TYPE_NAME

This method is used to return the technical name of the source entity type.

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_SOURCE_ENTITY_SET_NAME

This method is used to return the technical name of source entity set

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_SOURCE_KEYS

This method is used to return the table of the target entity keys.

Parameter	Description
RV_SOURCE_KEYS	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_PAIR is used to return the table of the source entity keys.

Method GET_NAVIGATION_PATH

This method is used to return the navigation path using technical names.

Parameter	Description
RT_NAVIGATION_PATH	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_NAVI is used to return the navigation path using technical names.

1.1.5.2.1.12 /IWBEF/IF_MGW_REQ_ENTITY_U

Use

This interface is developed to provide all request information relevant for entity UPDATE operation. All artifacts are returned in a technical format, meaning with their technical naming.

Methods

Method GET_ENTITY_TYPE_NAME

This method is used to return the technical name of the target entity type

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_ENTITY_SET_NAME

This method is used to return the technical name of target the entity set.

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_KEYS

This method is to be used when the entity is called directly: Source entity type and target entity type are the same. This method is used to return the table of keys for the requested entity. The returned key values are not converted even if they are defined in the model provider class as convertible. Use the new method GET_CONVERTED_KEYS to get the key values converted automatically.

Parameter	Description
RV_KEYS	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_PAIR is used to return the table of the target entity keys.

Method GET_CONVERTED_KEYS

This method is to be used when the entity is called directly: Source entity type and target entity type are the same. Data of type "Entity Return Structure" can be used because it already contains all keys and reference fields.

If a convertible field is used as key and the reference field for currency or quantity is needed, the reference field must also be defined as key. Otherwise, the returned key value cannot be converted and remains unchanged.

If a key field is missing in structure ES_KEY_VALUES the key value will not be provided to the provider application.

If one or more keys are defined in the model provider class as not convertible, the returned key values are not converted. Therefore, this method can be used in general instead of GET_KEYS.

Parameter	Description
ES_KEY_VALUES	This parameter of type DATA. This parameter must be a structure which contains all defined keys including the reference fields for currency or quantity if necessary that is, key value is an amount).

Method GET_CONDITIONAL_INFO

This method returns conditional/ETag information provided within the OData request.

The return type structure /IWBEF/IF_MGW_APPL_TYPES=>TY_S_CONDITIONS is defined as follows:

- IF_MATCH - contains the list of If-Match tags defined as the structure type /IWBEF/IF_MGW_APPL_TYPES=>TY_S_ETAG.
- IF_NONE_MATCH - contains the list of If-None-Match tags as of the structure type /IWBEF/IF_MGW_APPL_TYPES=>TY_S_ETAG.

The method is to be used if the conditional/ETag handling needs to be done by the application and not by the SAP Gateway framework.

The prerequisite to override the framework implementation is the implementation of the interface method

/IWBEF/IF_MGW_APPL_SRV_RUNTIME~GET_IS_CONDITIONAL_IMPLEMENTED.

Parameter	Description
ES_CONDITIONS	This parameter is of type /IWBEF/IF_MGW_APPL_TYPES=>TY_S_CONDITIONS.

1.1.5.2.1.13 /IWBEF/IF_MGW_REQ_ENTITY_D

Use

This Interface is developed to provide all request information relevant for entity DELETE operation. All artifacts are returned in a technical format meaning with their technical naming.

Methods

Method GET_ENTITY_TYPE_NAME

This method is used to return the technical name of the target entity type

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_ENTITY_SET_NAME

This method is used to return the technical name of target the entity set.

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_KEYS

This method is used to return the table of the target entity keys.

Parameter	Description
RV_KEYS	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_PAIR is used to return the table of the target entity keys.

1.1.5.2.1.14 /IWBEF/IF_MGW_REQ_ENTITY_P

Use

This Interface provides all request information relevant for an entity PATCH operation. Details for UPDATE related information can be found in [/IWBEF/IF_MGW_REQ_ENTITY_U](#). All artifacts are returned in a technical format, that is, with their technical naming.

Method GET_COMPONENTS

This method returns patched components.

Parameter	Description
ET_COMPONENTS	This parameter is of the type /IWBEP/IF_MGW_APPL_TYPES=>TY_T_COMPONENTS which contains all patched components in the PATCH request in a nested table if necessary (in case of complex properties).

1.1.5.2.1.15 /IWBEP/IF_MGW_REQ_ENTITY

This interface is developed to provide all request information relevant for entity READ operation. All artifacts are returned in a technical format meaning with their technical naming.

Methods

Method GET_ENTITY_TYPE_NAME

This method is used to return the technical name of the target entity type

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_ENTITY_SET_NAME

This method is used to return the technical name of target the entity set.

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_IF_MODIFIED_SINCE

This method is used to get the IF_MODIFIED_SINCE request header attribute.

Table 1:

Parameter	Description
RV_IF_MODIFIED_SINCE	This parameter is of type TZNSTMP.

Method GET_SOURCE_ENTITY_TYPE_NAME

This method is used to return the technical name of the source entity type.

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_SOURCE_ENTITY_SET_NAME

This method is used to return the technical name of source entity set

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_SOURCE_KEYS

This method is to be used when the entity is called by a navigation: Source entity type and target entity type are different. The method returns a table of source entity keys. The returned key values are not converted even if they are defined in the Model Provider Class as convertible. Use method GET_CONVERTED_SOURCE_KEYS to get the key values converted automatically.

Parameter	Description
RV_SOURCE_KEYS	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_PAIR is used to return the table of the source entity keys.

Method GET_KEYS

This method is to be used when the entity is called directly: Source entity type and target entity type are the same. The method returns a table of keys for the requested entity. The returned key values are not converted even if they are defined in the Model Provider Class as convertible. Use method GET_CONVERTED_KEYS to get the key values converted automatically.

Parameter	Description
RV_KEYS	This parameter of the type

/IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_PAIR is used to return the table of the source entity keys.

Method GET_NAVIGATION_PATH

This method is used to return the navigation path using technical names.

Parameter	Description
RT_NAVIGATION_PATH	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_NAVI is used to return the navigation path using technical names.

Method GET_SELECT

1 Note

This method is deprecated. Use method GET_SELECT_WITH_MANDTRY_FIELDS instead.

Select the system query option (\$select) to select only a subset of properties which should be returned by the server. The \$select can be considered by the application for performance reasons and improvements. The SAP Gateway runtime takes care of applying the \$select properly irrespective of whether \$select is taken into account from the application or not. So, for the consumer it is always the same behavior. It is only possible to specify a \$select query for simple properties or for complex properties of the entity type itself. Note that it is not possible to select properties of complex properties.

Method GET_SELECT_WITH_MANDTRY_FIELDS

Select the system query option (\$select) to select only a subset of properties which should be returned by the server. The \$select can be considered by the application for performance reasons and improvements. The SAP Gateway runtime takes care of applying the \$select properly irrespective of whether \$select is taken into account from the application or not. So, for the consumer it is always the same behavior. The SAP Gateway runtime always adds the mandatory fields of the entity type (keys, etag, mime type mapping) to the select table, irrespective of whether the key fields are already part of the original OData request or not. Therefore it is ensured that the result of this method can under certain circumstances, for example, be directly applied to an SQL statement without any enrichment from the application developer. It is only possible to specify a \$select query for simple properties or for complex properties of the entity type itself according to the OData specification. Note that it is not possible to select properties of complex properties.

In case of \$expand the result also contains the selected properties of all included expands. The selected properties of all included expands are necessary for service providers that implement their own expand instead of framework expand to improve performance.

If you only want to have the selected properties including the mandatory fields you use method GET_SELECT_ENTITY_PROPERTIES.

If \$select is not used the returned table RT_SELECT will be empty, and this means all entity properties are required.

When implementing your own expand (method GET_EXPANDED_ENTITY or GET_EXPANDED_ENTITYSET) you can use parameter IO_EXPAND to get the complete expand tree and all necessary information to process the whole expand completely or just partly to improve performance. See also [IF_MGW_ODATA_EXPAND](#).

Method GET_CONVERTED_SOURCE_KEYS

This method is to be used when the entity is called by a navigation: Source entity type and target entity type are different. ES_KEY_VALUES must be a structure which contains all defined keys including the reference fields for currency or quantity if necessary (that is, key value is an amount). Data of type "Entity Return Structure" can be used because it already contains all keys and reference fields. When convertible field is used as key and the reference field for currency or quantity is needed, the reference field must also be defined as key. Otherwise, the returned key value cannot be converted and remains unchanged.

If a key field is missing in structure ES_KEY_VALUES the key value will not be provided to the provider application. If one or more keys are defined in the Model Provider Class as not convertible the returned key values are not converted. Therefore, this method can be used in general instead of GET_SOURCE_KEYS.

Method GET_CONVERTED_KEYS

This method is to be used when the entity is called directly: Source entity type and target entity type are the same. ES_KEY_VALUES must be a structure which contains all defined keys including the reference fields for currency or quantity if necessary (that is, key value is an amount). Data of type "Entity Return Structure" can be used because it already contains all keys and reference fields. When convertible field is used as key and the reference field for currency or quantity is needed, the reference field must also be defined as key. Otherwise, the returned key value cannot be converted and remains unchanged.

If a key field is missing in structure ES_KEY_VALUES the key value will not be provided to the provider application. If one or more keys are defined in the Model Provider Class as not convertible the returned key values are not converted. Therefore, this method can be used in general instead of GET_KEYS.

Method GET_SELECT_ENTITY_PROPERTIES

Contrary to GET_SELECT_WITH_MANDTRY_FIELDS, this method only returns the selected properties including the mandatory fields of the current entity.

1.1.5.2.1.16 /IWBEF/IF_MGW_REQ_ENTITYSET

Use

This Interface represents the request attributes of an entityset request. All entity names used in the request object are the technical names of the entities, for example the names of the properties in the underlying data structures.

Methods

Method GET_ENTITY_TYPE_NAME

This method is used to return the technical name of the target entity type

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEF/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_ENTITY_SET_NAME

This method returns the technical name of the target entity set.

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_FILTER

This method is used to return the instance of the filter object.

Parameter	Description
RO_FILTER	This parameter of the type /IWBEP/IF_MGW_REQ_FILTER is used to return the technical name of the target entity type.

Method GET_IF_MODIFIED_SINCE

This method is used to get the IF_MODIFIED_SINCE request header attribute.

Table 1:

Parameter	Description
RV_IF_MODIFIED_SINCE	This parameter is of type TZNSTMP.

Method GET_SOURCE_ENTITY_TYPE_NAME

This method is used to return the technical name of the source entity type.

Parameter	Description
RV_ENTITY_TYPE	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity type.

Method GET_SOURCE_ENTITY_SET_NAME

This method is used to return the technical name of source entity set

Parameter	Description
RV_ENTITY_SET	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_NAME is used to return the technical name of the target entity set.

Method GET_SOURCE_KEYS

This method is to be used when the entity is called by a navigation: Source entity type and target entity type are different. This method returns a table of source entity keys. The returned key values are not converted even if they are defined in the model provider class as convertible. Use method GET_CONVERTED_SOURCE_KEYS to get the key values converted automatically.

Parameter	Description
RV_SOURCE_KEYS	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_E_TECHNICAL_PAIR is used to return the table of the source entity keys.

Method GET_NAVIGATION_PATH

This method is used to return the navigation path using technical names.

Parameter	Description
RT_NAVIGATION_PATH	This parameter of the type /IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_NAVI is used to return the navigation path using technical names.

Method GET_TOP

This method is used to get the paging information of the \$top parameter. The \$top parameter indicates the number of entries which should be returned by the server.

Parameter	Description
RV_TOP	This parameter has string which is only filled in case of \$top is given in OData request.

Method GET_SKIP

This method is used to get the paging information of the \$skip parameter. The \$skip parameter indicates the number of entries which should be skipped by the server.

Parameter	Description

RV_SKIP	This parameter has integer to specify the number of items which should be skipped.
---------	--

Method HAS_INLINECOUNT

If an inlinecount is requested the response has to contain a count of entries embedded in the payload. The count has to be calculated by the application and has to be passed via the response context.

Parameter	Description
RV_HAS_INLINECOUNT	This parameter contains a boolean value which indicates whether an inlinecount is requested by the client.

Method HAS_COUNT

This method indicates whether the client requested a count only.

If a count is requested the response contains the amount of entries only. The count can be calculated by the application and can be passed via the response context. If the counting is not done by the application the runtime derives the amount of entries via the returned data table.

Parameter	Description
RV_HAS_COUNT	This parameter contains a boolean value which indicates whether a count is requested by the client.

Method GET_SKIPTOKEN

The server can limit the amount of entries sent in a response in order to avoid a performance overhead if a huge number of entries is requested. A next link in the payload contains the skiptoken set by the server in the response context.

Parameter	Description
RV_SKIP	This parameter contains skiptoken which is generated by the server and used to identify the next entries which need to be sent back.

Method GET_DELTATOKEN

This method provides the delta token from a delta link within a feed. The delta token has previously been set in the response structure of a get feed call.

Note that a delta token needs to be issued in a normal feed to signal to the client that it supports delta tokens.

The way the delta token is issued and constructed is done by the issuer and is of his choice. For convenience sake you can use a date-time token (for default cases). In case of a date-time token you only need to take into account that the delta token should be related to one specific time zone, such as UTC.

A delta token call from a client needs to be handled appropriately. The delta token holds specific information which is known by the issuer. With this information it should be easy to extract the changed data. Deleted records are not supported at present.

Parameter	Description
RV_DELTATOKEN	This parameter contains the delta token which is generated by the server and is used to identify the delta entries which need to be sent back.

Method GET_SEARCH_STRING

If a search is used then this method contains the search string.

A search string can be applied to a Collection request which can be used as a freetext parameter.

Parameter	Description
RV_SEARCH_STRING	This parameter contains the search parameter provided by the client.

Method GET_SELECT

Note

This method is deprecated. Use `GET_SELECT_WITH_MANDTRY_FIELDS` instead.

Select **System Query Option (\$select)** to select only a subset of properties which should be returned by the server. The \$select can be considered by the application for performance reasons and improvements. The SAP Gateway runtime takes care of applying the \$select properly irrespective of whether \$select is taken into account from the application or not. So for the consumer, it is always the same behavior. It is only possible to specify a \$select query for simple properties or for complex properties of the Entity Type itself. Note that it is not possible to properties of complex properties.

Parameter	Description
RT_SELECT	This parameter of type <code>/IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_NAME</code> contains the system query option (\$select).

GET_SELECT_WITH_MANDTRY_FIELDS

Select the system query option (\$select) to select only a subset of properties which should be returned by the server. The \$select can be considered by the application for performance reasons and improvements. The SAP Gateway runtime takes care of applying the \$select properly irrespective of whether \$select is taken into account from the application or not. So, for the consumer it is always the same behavior. The SAP Gateway runtime always adds the mandatory fields of the entity type (keys, etag, mime type mapping) to the select table irrespective of whether the key fields are already part of the original OData request or not. Therefore it is ensured that the result of this method can, for example, directly be applied to an SQL statement without any enrichment from the application developer.

It is only possible to specify a \$select query for simple properties or for complex properties of the entity type itself according to the OData specification. Note that it is not possible to select properties of complex properties.

In case of \$expand the result also contains the selected properties of all included expands. The selected properties of all included expands are necessary for service providers that implement their own expand instead of framework expand to improve performance.

If you only want to have the selected properties including the mandatory fields you have to use `GET_SELECT_ENTITY_PROPERTIES`.

If \$select is not used the returned table `RT_SELECT` will be empty, and this means all entity properties are required.

When implementing your own expand (method `GET_EXPANDED_ENTITY` or `GET_EXPANDED_ENTITYSET`) you can use the parameter `IO_EXPAND` to get the complete expand tree and all necessary information to process the whole expand completely or just partly to improve performance. See also [/IWBEP/IF_MGW_ODATA_EXPAND](#).

GET_ORDERBY

OrderBy system query option (\$orderby) .

Parameter	Description
RT_ORDERBY	This parameter of type <code>/IWBEP/IF_MGW_CORE_SRV_RUNTIME=>TY_T_TECHNICAL_ORDER</code> contains the OrderBy system query option (\$orderby).

Method GET_CONVERTED_SOURCE_KEYS

This method is to be used when the entity set is called by a navigation: Source entity type and target entity type are different. `ES_KEY_VALUES` must be a structure which contains all defined keys including the reference fields for currency or quantity if necessary (that is, key value is an amount). Data of type "Entity Return Structure" can be used because it already contains all keys and reference fields. If a convertible field is used as key and the reference field for currency or quantity is needed, the reference field must also be defined as key. Otherwise, the returned key value cannot be converted and remains unchanged. If a key field is missing in structure `ES_KEY_VALUES` the key value will not be provided to the provider application. If one or more keys are defined in the Model Provider Class as not convertible the returned key values are not converted. Therefore, this method can be used in general instead of `GET_SOURCE_KEYS`.

Table 2:

Parameter	Description
ES_KEY_VALUES	Type DATA

Method GET_FILTER_EXPRESSION_TREE

This method is to be used to get the filter expression tree.

Parameter	Description
RO_FILTER_TREE	This parameter contains \$filter system query option in a object tree format.

Method GET_OSQSQL_WHERE_CLAUSE

This method is to be used to get an Open SQL WHERE clause.

Parameter	Description
RV_OSQSQL_WHERE_CLAUSE	This parameter contains the Open SQL WHERE Clause string generated from \$filter system query option.

Method GET_SELECT_ENTITY_PROPERTIES

Contrary to `GET_SELECT_WITH_MANDTRY_FIELDS`, this method only returns the selected properties including the mandatory fields of the current entity.

Parameter	Description
RT_SELECT	Type STRING_TABLE

Method IS_CACHE_PAGE_ON_HB_ALLOWED

The method returns `TRUE` if the soft state based query result cache (SQRC) of the SAP Gateway framework can be used. That is, the functionality to "cache and page on hub" can be used. This is the case if:

- Soft state for the service has been activated
- and if the SQRC has not been de-activated for the service by the system administrator via transaction `/IWBEP/CONF_SERVICE`

If this method returns `TRUE` you can set the exporting parameter `ES_RESPONSE_CONTEXT- DO_CACHE_AND_PAGE_ON_HUB` of method `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET` to `TRUE` as well. In this case the soft state based query result cache (SQRC) of the SAP Gateway framework is activated.

Parameter	Description
RV_IS_CACHE_AND_PAGE_OK	Type ABAP_BOOL

SQRC Functionality

A GET for an entity set is requested by a client. This will be handled by the SQRC if the following conditions apply:

- Soft state for the service has been enabled in transaction `/IWFND/MAINT_SERVICE`
This is only possible if the corresponding DPC (data provider class) redefines the methods of interface `/IWBEP/IF_MGW_CORE_SRV_RUNTIME`. Software components `IW_BEP` and `IW_FND` have to be on support package stack 09 or SAP NetWeaver 7.40 SP08 (SAP_GWEND) respectively.
- Your data provider in method `/IWBEP/IF_MGW_APPL_SRV_RUNTIME~GET_ENTITYSET` does the following:
 - It applies the filter (system query option \$filter)
 - It applies the sorting (system query option \$sort)
 - It does not apply the paging (system query options \$top and \$skip)
 - It returns `ES_RESPONSE_CONTEXT-DO_CACHE_AND_PAGE_ON_HUB = ABAP_TRUE` together with the list of entities
 - It does not do server-side paging (triggered via `ES_RESPONSE_CONTEXT-SKIPTOKEN` or `ES_RESPONSE_CONTEXT-EXPAND_SKIPTOKEN`)

- Consecutive client requests are a GET on an Entity Set.
The only difference between the request URLs are the values of the \$top and \$skip system query options. The rest of the URL must be identical.

1.1.5.2.1.17 /IWBEP/IF_MGW_EXPR_VISITOR

Use

This is a visitor interface which can be implemented to process the filter expression tree.

Methods

Method PROCESS_LITERAL

This method processes the literal expression of the filter expression tree.

Parameter	Description
IO_LITERAL	This parameter is of type /IWBEP/IF_MGW_EXPR_LITERAL which contains literal expression.

Method PROCESS_PROPERTY

This method processes the property expression of the filter expression tree.

Parameter	Description
IO_PROPERTY	This parameter is of type /IWBEP/IF_MGW_EXPR_PROPERTY which contains property expression.

Method PROCESS_UNARY

This method processes the unary expression of the filter expression tree.

Parameter	Description
IO_UNARY	This parameter is of type /IWBEP/IF_MGW_EXPR_UNARY which contains unary expression.

Method PROCESS_BINARY

This method processes the binary expression of the filter expression tree.

Parameter	Description
IO_BINARY	This parameter is of type /IWBEP/IF_MGW_EXPR_BINARY which contains binary expression.

Method PROCESS_FUNCTION

This method processes the function expression of the filter expression tree.

Parameter	Description
IO_FUNCTION	This parameter is of type /IWBEP/IF_MGW_EXPR_FUNCTION which contains function expression.

Method PROCESS_MEMBER

This method processes the member expression of the filter expression tree.

Parameter	Description
IO_MEMBER	This parameter is of type /IWBEP/IF_MGW_EXPR_MEMBER which contains member expression.

1.1.5.2.2 OData Channel Metadata APIs

Describes the central OData Channel metadata classes and interfaces.

Use

The central OData Channel metadata classes and interfaces are as follows:

- [/IWBEP/CL_MGW_ABS_MODEL](#)
- [/IWBEP/IF_MGW_MED_ODATA_TYPES](#)
- [/IWBEP/IF_MGW_ODATA_ACTION](#)
- [/IWBEP/IF_MGW_ODATA_ANNOTATABL](#)
- [/IWBEP/IF_MGW_ODATA_ANNOTATION](#)
- [/IWBEP/IF_MGW_ODATA_ASSOC](#)
- [/IWBEP/IF_MGW_ODATA_ASSOC_SET](#)
- [/IWBEP/IF_MGW_ODATA_CMLPX_PROP](#)

- [/IWBEP/IF_MGW_ODATA_CMPLX_TYPE](#)
- [/IWBEP/IF_MGW_ODATA_DOCUMENTTTN](#)
- [/IWBEP/IF_MGW_ODATA_ENTITY_SET](#)
- [/IWBEP/IF_MGW_ODATA_ENTITY_TYP](#)
- [ABAP Dictionary Type to EDM.Type Mapping](#)
- [/IWBEP/IF_MGW_ODATA_ITEM](#)
- [/IWBEP/IF_MGW_ODATA_MODEL](#)
- [/IWBEP/IF_MGW_ODATA_NAV_PROP](#)
- [/IWBEP/IF_MGW_ODATA_PARAMETER](#)
- [/IWBEP/IF_MGW_ODATA_PROPERTY](#)
- [/IWBEP/IF_MGW_ODATA_REF_CONSTR](#)

Original Names

With SAP Gateway 2.0 SP05 the **redefine function** is supported. This means that after the definition of a model artifact it can be renamed.

During the model definition (`/IWBEP/CL_MGW_ABS_MODEL->DEFINE ()`) a model provider always needs to use the original name when referencing an artifact. This is independent of the fact that it might already have been renamed.

Example

```

        lo_entity_type = model->create_entity_type(
            iv_entity_type_name = gcs_entity_names-employee
        ).
lo_entity_type->set_name( 'Worker' ).                "Some renaming...
...
lo_action->set_action_for( gcs_entity_names-employee ).    "Using the original name

```

Technical Names

A data provider should only use the import parameter `IO_TECH_REQUEST_CONTEXT`. That one "speaks" the technical (internal) names and hence is not influenced by any redefine function. The internal names - unless explicitly set by the model provider - are derived from the "original" name.

1.1.5.2.2.1 /IWBEP/CL_MGW_ABS_MODEL

This is the abstract class for a model provider class (MPC).

This is the abstract class for a model provider class (MPC). Every concrete implementation of a model provider class shall be derived from this class.

Several constants are defined for this class.

Constant	Description
GCS_SAP_TEXT_OBJECT_TYPES	Constants can be use to address the kind of text which can be used, for example a text symbol of a class in the backend. The structure contains a list of the possible text object types. This text object should be used to create text with method SET_TEXT_KEY for a field label. <code>TEXT_SYMBOL_CLASS</code> uses a text element of a class in the backend system.
CARDINALITY_FEED	Value=* for multiplicity / cardinality of an association end type, for example.
CARDINALITY_ENTITY	Value=1 for multiplicity / cardinality of an association end type, for example.
CARDINALITY_ENTITY_0	Value=0..1 for multiplicity / cardinality of an association end type, for example.
GCS_SAP_SEMANTIC	All possible values as constants for annotation <code>sap:semantics</code> .
GCS_FC_TARGET_PATH	Values for target path of feed customization, for example for content source and type of a Media Links entry.

There is only one **member**, that is the `MODEL` which is the starting object to define the OData Channel data model.

Methods

Method DEFINE

This method needs to be implemented by an application to define its metadata. The entry point is the member data model.

Method GET_LAST_MODIFIED

Based on the implementation of this method a time stamp is sent in the HTTP response header of every metadata request to give the Web infrastructure the ability to cache the metadata document.

Note

The default implementation of this method derives the last modified time stamp based on the changed time stamp of the model provider class (MPC). The application can overwrite the logic if it does not fit.

Parameter	Description
RV_LAST_MODIFIED	Time stamp which is set as Last-Modified in the HTTP response header.

Starting with SAP Gateway 2.0 SP06 (`IW_BEP` and `IF_FND`) the result of `GET_LAST_MODIFIED` also controls the cache on the SAP Gateway server for service document and service metadata document requests. If the SAP Gateway metadata cache is active, then the SAP Gateway server performs a handshake with the backend, based on the `GET_LAST_MODIFIED` result and compares the timestamp from the cached metadata on the SAP Gateway server with the one returned by the `GET_LAST_MODIFIED` implementation of the Model Provider on the backend.

If the cache is outdated on the SAP Gateway server, then the new metadata is sent from the backend and the cache is updated with the new metadata on the SAP Gateway server. As a result, the latest changes of your metadata is always reflected when requesting a service and a service metadata document. If the cache is not outdated on the SAP Gateway server, then the backend does not send any metadata back to the SAP Gateway server, because the cached metadata is still up-to-date.

Note

As a result, the latest changes of your metadata is always reflected when requesting a service and a service metadata document.

Caution

With this behavior the following setup will lead to a performance bottleneck for service and service metadata document requests, because the cache is in this case refreshed in every request for metadata:

- Method `GET_LAST_MODIFIED` is overwritten with coding like `GET TIME STAMP FIELD rv_last_modified`.
- The SAP Gateway metadata cache is active.

In order to always have the latest metadata present on your client applications, it is recommended to perform a request to the service metadata document as a very first call of your client application choreography (for example when starting the application) to the SAP Gateway server.

Method `GET_VOCAN_TEXTS`

The method is called with a table of all annotation text keys that have been defined in the corresponding model.

The framework implementation does the following:

- It loads the corresponding texts for all referenced data elements and referenced text elements.
- It calls method `GET_VOCAN_PROVIDER_TEXTS` to retrieve the provider texts in a second step.

As a rule there is no need to overwrite this method.

Parameter	Description
<code>IV_LANGUAGE</code>	The current language.
<code>IT_VOCAN_TEXT_KEYS</code>	Table of all annotation text keys that have been defined in the corresponding model.
<code>CT_VOCAN_TEXTS_OBJ</code>	Table containing the actual annotation texts.

Method `GET_VOCAN_PROVIDER_TEXTS`

The method is called with a table of annotation text keys that have been defined in the corresponding model, but only for the text type "provider text".

The framework implementation simply throws a `NOT-IMPLEMENTED` exception. A provider using its own text repository must overwrite this method and return the texts for all provider texts that have been defined in the model and which are also provided via import parameter `IT_VOCAN_TEXT_KEYS`.

Parameter	Description
<code>IV_LANGUAGE</code>	The current language.
<code>IT_VOCAN_TEXT_KEYS</code>	Table of annotation text keys, of the type "provider text".
<code>CT_VOCAN_TEXTS_OBJ</code>	Table containing the actual annotation texts.

1.1.5.2.2.2 /IWBEP/IF_MGW_MED_ODATA_TYPES

Use

This is the central type interface for OData Channel.

1.1.5.2.2.3 /IWBEP/IF_MGW_ODATA_ACTION

Use

This interface represents an OData service operation / function import: see <http://www.odata.org/developers/protocols/operations#InvokingServiceOperations> and <http://www.odata.org/developers/protocols/uri-conventions#AddressingServiceOperations>.

Methods

Method `SET_RETURN_ENTITY_TYPE`

This method sets the return entity type of a function import.

Note

Note that a function import can either return an entity type or a complex type.

Parameter	Description
<code>IV_DATA_OBJECT_NAME</code>	Name of the entity type which is returned by the function import and which is declared for the current model.

Method SET_RETURN_ENTITY_SET

This method sets return entity set of a function import which can be a single entity set or a collection of entity sets depending on the defined multiplicity.

Note

Note that a function import can either return an entity type or a complex type.

Parameter	Description
IV_ENTITY_SET_NAME	Name of the entity set which is returned by the function import and which is declared for the current model.

Method SET_RETURN_MULTPLICITY

This method sets multiplicity to indicate whether a function import returns a collection / feed of entries or just a single entry.

Parameter	Description
IV_MULTPLICITY	Use constants <code>/IWBEP/CL_MGW_ABS_MODEL=>CARDINALITY_FEED</code> or <code>/IWBEP/CL_MGW_ABS_MODEL=>CARDINALITY_ENTITY</code> to set the multiplicity.

Method SET_HTTP_METHOD

This method sets the HTTP method of a function import, for example GET.

The HTTP method of a function import is used to indicate whether the function import does any modifications or is read-only. Modifying function imports should use PUT, non-modifying should set GET.

Parameter	Description
IV_HTTP_METHOD	Method, for example GET or POST.

Method CREATE_INPUT_PARAMETER

This method defines an input parameter for the function import.

Note

Note that parameters are only passed via URL parameters and not in the request body (OData 2.0 Specification).

Parameter	Description
IV_PARAMETER_NAME	External parameter name which is shown in the service metadata document and used when invoking a service operation.
IV_ABAP_FIELDNAME	Internal ABAP name representation of the parameter.
RO_PARAMETER	Instance of parameter to do a data element binding, for example.

Method SET_RETURN_COMPLEX_TYPE

This method sets the return complex type of a function import.

Note

Note that a function import can either return an entity type or a complex type. In SAP Gateway it is only possible to have single complex types as return types. Collections are not allowed and therefore multiplicity had best use `/IWBEP/CL_MGW_ABS_MODEL=>CARDINALITY_ENTITY` for return complex types.

Parameter	Description
IV_COMPLEX_TYPE_NAME	Name of the complex type which is declared in the model and used as return type.

Method SET_ACTION_FOR

This method sets the value of the sap annotation `sap:action-for`.

Parameter	Description
IV_ENTITY_TYPE_NAME	Name for which the action can be applied.

1.1.5.2.2.4 /IWBEP/IF_MGW_ODATA_ANNOTATABL

Use

This interface is implemented by every OData Channel entity to add additional annotation attributes.

Method CREATE_ANNOTATION

This method creates an annotation for the OData Channel entity.

Parameter	Description
IV_ANNOTATION_NAMESPACE	Namespace which is used for the annotation, for example sap.

RO_ANNOTATION	Instance to set all required parameters.
---------------	--

/IWBEP/IF_MGW_ODATA_ANNOTATION

Use

Interface of an annotation which is returned when adding an annotation attribute to an OData Channel entity.

Methods

Method ADD

This method specifies key and value of the annotation for the actual OData Channel entity.

Parameter	Description
IV_KEY	Key of an annotation attribute, for example label in namespace <code>sap</code> in the service metadata document.
IV_VALUE	Value of an annotation attribute, for example value of label in namespace <code>sap</code> in the service metadata document.

Method ADD_CHILD

This method specifies key and value of an annotation child attribute for the actual OData Channel entity.

Parameter	Description
IV_PARENT_KEY	Annotation key.
IV_KEY	Key of an annotation child attribute.
IV_VALUE	Value of an annotation child attribute.

Note

The annotations you specify are used for documentation purposes only and are not otherwise evaluated. That is, they simply provide additional descriptive information.

1.1.5.2.2.6 /IWBEP/IF_MGW_ODATA_ASSOC

Use

Interface which represents an OData association. In the following, some of the methods are described.

Methods

Method CREATE_REF_CONSTRAINT

This method returns a referential constraint object to map to entity types and specify the foreign key relationship.


Parameter	Description
IV_PRINCIPAL_IS_LEFT	
RO_REF_CONSTRAINT	Instance to define a referential constraint.

Method CREATE_ASSOC_SET

This method creates an AssociationSet for the association.

Parameter	Description
IV_ASSOC_SET_NAME	AssociationSet name which is part of the service metadata document.
RO_ASSOC_SET	Instance to set further parameters of the AssociationSet.

SET_LEFT_ON_DELETE_ACTION/SET_RIGHT_ON_DELETE_ACTION

These methods are available with [2272007](#)  and provide information for the client about deletion behavior. The actions `none` and `cascade` can be set for both ends of an association.

For example, when deleting the entityType `SalesOrder` which has an association to the entityType `SalesOrderItem`, should the `SalesOrderItem` also be deleted automatically. In case of `none`, no `SalesOrderItem` will be deleted, and in case of `cascade`, the `SalesOrderItems` will be deleted.

Parameter	Description
IV_ON_DELETE_ACTION	

1.1.5.2.2.7 /IWBEP/IF_MGW_ODATA_ASSOC_SET

Use

Interface which represents an OData AssociationSet.

1.1.5.2.2.8 /IWBEP/IF_MGW_ODATA_CMPLX_PROP

This interface represents an OData complex property object based on a complex Type.

Method GET_COMPLEX_TYPE

This method returns the complex type on which this property is based.

Parameter	Description
RO_COMPLEX_TYPE	Complex type

1.1.5.2.2.9 /IWBEP/IF_MGW_ODATA_CMPLX_TYPE

Use

This interface represents an OData complex type object.

Methods

Method BIND_STRUCTURE

See [/IWBEP/IF_MGW_ODATA_ENTITY_TYP](#) method `BIND_STRUCTURE`.

Method CREATE_COMPLEX_PROPERTY

Note

Note that this method is obsolete.

This is a method for nested complex types.

Parameter	Description
IV_PROPERTY_NAME	Name in the complex type.
IV_COMPLEX_TYPE_NAME	Name of the complex type which is added as a nested complex type.
IV_ABAP_FIELDNAME	Field name of the complex type, property.

Method CREATE_CMPLX_TYPE_PROPERTY

This is a method for nested complex types.

Parameter	Description
IV_PROPERTY_NAME	Name in the complex type.
IV_COMPLEX_TYPE_NAME	Name of the complex type which is added as a nested complex type.
IV_ABAP_FIELDNAME	Field name of the complex type, property.
RO_COMPLEX_PROP	Instance of the new complex type property of type <code>IF_MGW_ODATA_CMPLX_PROP</code> .

Method CREATE_PROPERTY

This method adds a property to the complex type.

Parameter	Description
IV_PROPERTY_NAME	Name in the complex type.
IV_ABAP_FIELDNAME	Field name of the complex type, property.
RO_PROPERTY	Instance.

Method GET_PROPERTY

This method retrieves a property of the complex type.

Parameter	Description
IV_PROPERTY_NAME	Name in the complex type.
RO_PROPERTY	Instance.

Method SET_BASE_TYPE

This method retrieves a property of the complex type.

Parameter	Description
IV_PROPERTY_NAME	Name in the complex type.
RO_PROPERTY	Instance.

Method SET_IS_ABSTRACT

This method sets the complex type to abstract.

Parameter	Description
IV_IS_ABSTRACT	Boolean value which indicates whether the complex type can be instantiated or not.

Method ADD_AUTO_EXPAND_INCLUDE

See method `ADD_AUTO_EXPAND_INCLUDE` of interface `/IWBEP/IF_MGW_ODATA_ENTITY_TYP`.

1.1.5.2.2.10 /IWBEP/IF_MGW_ODATA_DOCUMENTTTN

Use

This interface represents an OData Channel documentation object.

Method SET_SUMMARY_FROM_TEXT_ELEMENT

This method creates a short text describing the artifact. The text is referred to by a text key in the model and is loaded, depending on the language, after the metadata.

Parameter	Description
IV_TEXT_ELEMENT_SYMBOL	Text element key
IV_TEXT_ELEMENT_CONTAINER	Text element container
IO_OBJECT_REF	Object reference

Method SET_LONGTEXT_FROM_TEXT_ELEMENT

This method creates a detailed description of the artifact. The text is referred to by a text key in the model and is loaded, depending on the language, after the metadata.

Parameter	Description
IV_TEXT_ELEMENT_SYMBOL	Text element key
IV_TEXT_ELEMENT_CONTAINER	Text element container
IO_OBJECT_REF	Object reference

1.1.5.2.2.11 /IWBEP/IF_MGW_ODATA_ENTITY_SET

Use

This is an interface which represents an OData Channel EntitySet object.

Methods

Method SET_TITLE

Note

This method is **deprecated**. Use method `SET_MEMBER_TITLE_FR_TELEMENT` instead.

The method creates a description for a field `<atom:title>` that differs from those of the ABAP Dictionary. A source of this text can be a text symbol of a class.

The semantics of the input parameter must be the `CLASS NAME` and the symbol ID of the text element separated by `-`. The corresponding text object key is defined in class `/IWBEP/CL_MGW_ABS_MODEL`.

Parameter	Description
IV_TEXT_KEY	Text key, for example, a reference to a text symbol from text references of a class or report. Example: <code>/IWBEP/CL_MGW_MED_SFLIGHT-000</code> The text is not case-sensitive.
IV_TEXT_OBJ_TYPE	Text object type. Example: <code>GCS_SAP_TEXT_OBJECT_TYPES-BACKEND_GATEWAY_TEXT</code>

Method SET_MEMBER_TITLE

Note

This method is **deprecated**. Use method `SET_MEMBER_TITLE_FR_TELEMENT` instead.

The method creates a description for a field `<sap:member-title>` that differs from those of the ABAP Dictionary. A source of this text can be a text symbol of a class.

The semantics of the input parameter must be the `CLASS NAME` and the symbol ID of the text element separated by `-`. The corresponding text object key is defined in class `/IWFBEF/CL_MGW_ABS_MODEL`.

Parameter	Description
IV_TEXT_KEY	Text key in the class name and the text element separated by <code>-</code> . Example: <code>/IWFBEF/CL_MGW_MED_POOL_TX_LOAD-000</code> The text is not case-sensitive.
IV_TEXT_OBJ_TYPE	Text in the text object type defined in the field mentioned above. Example: <code>GCS_SAP_TEXT_OBJECT_TYPES-BACKEND_GATEWAY_TEXT</code>

Method SET_MEMBER_TITLE_FR_TELEMENT

This method creates the `<sap:member-title>` of a collection from a reference to a text element.

Parameter	Description
IW_TEXT_ELEMENT_SYMBOL	Symbol of the text element, for example <code>000</code> .
IV_TEXT_ELEMENT_CONTAINER	Name of the text element container, for example, a report name or the class name. This is an optional parameter which should be filled if the text reference is located in another class or report and not the MPC.
IO_OBJECT_REF	Reference of the class which contains the text element. If the MPC contains the texts ABAP _{me} needs to be passed.

Method SET_CREATABLE

This method indicates whether instances of this type can be created. The default value is `true` and therefore not shown in the service metadata document.

Table 1:

Parameter	Description
IV_CREATABLE	Value of EntitySet

Method SET_UPDATABLE

This method indicates whether instances of this type can be updated. The default value is `true` and therefore not shown in the service metadata document.

Table 2:

Parameter	Description
IV_UPDATABLE	Value of EntitySet

Method SET_DELETABLE

This method indicates whether instances of this type can be deleted. The default value is `true` and therefore not shown in the service metadata document.

Table 3:

Parameter	Description
IV_DELETABLE	Value of EntitySet

Method SET_PAGEABLE

This method indicates whether paging can be applied to an EntitySet, for example parameters `$top` and `$skip`. The default value is `true` and therefore not shown in the service document for the collection (directly associated to EntitySet).

Table 4:

Parameter	Description
IV_PAGEABLE	Value

Method SET_ADDRESSABLE

This method indicates whether an EntitySet can be addressed directly or only via NavigationProperty. The default value is `true` and therefore not shown in the service document.

Table 5:

Parameter	Description
IV_ADDRESSABLE	Value

Method SET_SUBSCRIBABLE

This method indicates whether it is possible to subscribe to this type. The default value is `false` and therefore nothing is shown in the service document. Types supporting subscriptions are annotated with an `atom:link` element with attribute `rel="http://www.sap.com/Protocols/SAPData/rel#subscribe"` in the service document.

Table 6:

Parameter	Description
-----------	-------------

Parameter	Description
IV_IS_SUBSCRIBABLE	Value

Method SET_FILTER_REQUIRED

Method HAS_FTXT_SEARCH

Collections of this data type can be searched through OpenSearch. The default value is `false` and therefore nothing is shown in the service document. Types supporting OpenSearch are annotated an `atom:link` element with attribute `rel="search"` in the service document. The `atom:link` href points to the OpenSearch description document.


Table 7:

Parameter	Description
IV_FSEARCH	Value

For more information, see [Search and Open Search Capabilities](#).

1.1.5.2.2.12 /IWBEF/IF_MGW_ODATA_ENTITY_TYP

Use

This is an interface which represents an OData Channel EntityType object. See also <http://www.odata.org/developers/protocols/overview#EntityDataModel> .

Methods

Method BIND_STRUCTURE

This method binds the EntityType to an ABAP Dictionary structure or to a type definition in an ABAP interface, for example.

Every component of the structure which shall be exposed in the EntityType needs to be declared first as a property. The bind structure follows a white-list approach. Only properties which are explicitly declared for the EntityType are exposed. Structure binding is used to retrieve the type and the text information from the ABAP Dictionary. The EDM core types are automatically retrieved from the internal ABAP types but can be manually overwritten. The text information is retrieved based on the text descriptions stored for the data elements in the ABAP Dictionary.

See also [ABAP Dictionary Type to EDM.Type Mapping](#).

Parameter	Description
IV_STRUCTURE_NAME	Name of the structure data type.
IV_BIND_CONVERSIONS	Flag which propagates information about conversion exits, currency, and unit fields from the ABAP Dictionary into the metadata. The effect is the same as calling the methods <code>SET_CONVERSION_EXIT</code> , <code>SET_SEMANTIC</code> and <code>SET_UNIT</code> of the <code>/IWBEF/IF_MGW_ODATA_PROPERTY</code> interface. In addition for the amount, properties precision and scale are set according to the formula: Precision = Length * 2 - 1, Scale = Precision, where length is an internal length of the structure field.

Method CREATE_PROPERTY

This method adds a new property to the EntityType.

Parameter	Description
IV_PROPERTY_NAME	Name of the property.
IV_ABAP_FIELDNAME	Field name of the property.
RO_PROPERTY	Instance

Method GET_PROPERTY

This method retrieves the property of the EntityType by name (**not** the ABAP name).

Parameter	Description
IV_PROPERTY_NAME	Name of the property.
RO_PROPERTY	Instance

Method SET_CREATABLE

This method sets the value of `sap:creatable` of the EntityType. Thus it indicates whether instances of this type can be created. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_CREATABLE	Value of the EntityType.

Method SET_UPDATABLE

This method sets the value of `sap:updatable` of the EntityType. Thus it indicates whether instances of this type can be updated. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
-----------	-------------

IV_UPDATABLE	Value of the EntityType.
--------------	--------------------------

Method SET_DELETABLE

This method sets the value of `sap:deletable` of the EntityType. Thus it indicates whether instances of this type can be deleted. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_DELETABLE	Value of the EntityType.

Method SET_PAGEABLE

This method sets the value of `sap:pageable` of an EntitySet of this EntityType. Thus it indicates whether paging can be applied to an EntitySet of EntityType, for example parameters `$top` and `$skip`. The default value is `true` and therefore not shown in service document for the collection (directly associated to the EntitySet).

Parameter	Description
IV_PAGEABLE	Value

Method SET_ADDRESSABLE

This method sets the value of `sap:addressable` of an EntitySet of this EntityType. Thus it indicates whether the EntitySet of the EntityType can be addressed directly or only via navigation property. The default value is `true` and therefore not shown in service document.

Parameter	Description
IV_ADDRESSABLE	Value

Method SET_IS_MEDIA

This method sets the value of `m:HasStream` of an EntityType in the service metadata document. The default value is `false` if the method is not called and therefore not shown in service metadata document.

Instances of this type which are marked as media types are basically called media link entries. Every media link entry has an associated media resource which represents the binary which is referenced by the media link entry. Every instance of this type has a content source and a content MIME type (<http://www.odata.org/developers/protocols/atom-format#RepresentingMediaLinkEntries>). The following kind of media types are supported:

- Media types which point to external sources (external URLs)
 - A property is required in the EntityType which is marked as the content source via feed customization.
 - A property is required in the EntityType which is marked as the content MIME type via feed customization.
- Media types which point to internal sources (that is, binaries)
 - A property is required in the EntityType which is marked as the content MIME type via feed customization.
 - The content source shall not be set for internal sources. A content source link is generated by the framework and routes requests for media resources to the application specific implementation.

See also [Media Links](#).

Parameter	Description
IV_IS_MEDIA	Value

Method SET_IS_ABSTRACT

This method sets the abstract value for the EntityType.

Parameter	Description
IV_IS_ABSTRACT	Value

Method SET_SUBSCRIBABLE

This method indicates whether it is possible to subscribe to this type. The default value is `false` and therefore nothing is shown in the service document.

Types supporting subscriptions are annotated with an `atom:link` element with the attribute `rel="http://www.sap.com/Protocols/SAPData/rel#subscribe"` in the service document.

Parameter	Description
IV_IS_SUBSCRIBABLE	Value

Method SET_FILTER_REQUIRED

This method sets the flag that a filter is required for getting feeds.

Method HAS_FTXT_SEARCH

Collections of this data type can be searched through OpenSearch. The default value is `false` and therefore nothing is shown in the service document.

Types supporting OpenSearch are annotated with an `atom:link` element with attribute `rel="search"` in the service document. The `atom:link href` points to the OpenSearch description document.

Parameter	Description
IV_FSEARCH	Value

Method SET_SEMANTIC

This method sets the semantics of the entity type.

Supported values are the components of the structure `/IWBEP/CL_MGS_ABS_MODEL=>GCS_SAP_SEMANTIC`.

Parameter	Description
IV_SEMANTIC	Semantics in <code>/IWBEP/IF_MGW_MED_ODATA_TYPES=>TY_E_MED_SEMANTIC</code> .

Method SET_IS_THING_TYPE

This method represents an entity that defines tangible work objects. A "prominent" data object `EntityType` fulfilling additional constraints.

Parameter	Description
IV_IS_THING_TYPE	Value.

Method SET_BASE_TYPE

This method sets the base type of an `EntityType`.

Parameter	Description
IV_ENTITY_TYPE	Name of current type base.

Method CREATE_COMPLEX_PROPERTY

This method adds a complex property to the `EntityType`.

Parameter	Description
IV_PROPERTY_NAME	Property name in complex type.
IV_COMPLEX_TYPE_NAME	Complex type name of complex type which is added as a nested complex type.
IV_ABAP_FIELDNAME	Field name of the complex type, property.

Method CREATE_ENTITY_SET

This method defines an `EntitySet` which refers to the current `EntityType` object.

Parameter	Description
IV_ENTITY_SET_NAME	<code>EntitySet</code> name which appears, for example, in the service document as the href of the collection.
RO_ENTITY_SET	Instance

Method CREATE_NAVIGATION_PROPERTY

This method defines a navigation property.

Parameter	Description
IV_PROPERTY_NAME	Name of the navigation property.
IV_ASSOCIATION_NAME	Name which is referenced by the navigation property (relationship attribute of navigation property in Service Metadata Document).
IV_ABAP_FIELDNAME	Name of the navigation property (component name in case of expand or deep insert in expected structure)
RO_NAVIGATION_PROPERTY	Instance

Method GET_PROPERTIES

This method retrieves a list of all properties of the current `EntityType`.

Parameter	Description
RT_PROPERTIES	Properties of the <code>EntityType</code> .

Method SET_TEXT_KEY

This method sets the source of a text label, for example, a text element or ABAP Dictionary label.

The method create a description for a field that differs from those of the ABAP Dictionary. A source of this text can be a text symbol of a class. The semantics of the input parameter must be the `CLASS NAME` and the symbol Id of the text element separated by `-`. The corresponding text object key is defined in class `/IWBEP/CL_MGW_ABS_MODEL`.

Parameter	Description
IV_TEXT_KEY	Key in the class name and the text element separated by <code>-</code> . Example: <code>/IWFBEF/CL_MGW_MED_POOL_TX_LOAD-000</code>
IV_TEXT_OBJECT_TYPE	Type in the text object type defined in the field mentioned above. Example: <code>GCS_SAP_TEXT_OBJECT_TYPES-BACKEND_GATEWAY_TEXT</code>
IV_CREATE	In the text the key will be created if not available. This parameter must always be set to <code>ABAP_TRUE</code> .

Method ADD_AUTO_EXPAND_INCLUDE

This method adds an ABAP Dictionary structure to the entity type as an auto-expand include..

Simple field extensibility: the framework automatically and dynamically creates a corresponding entity property for each field of the so-called auto-expand include (except the dummy field) at runtime. Using this API the service provider developers does not need to modify the model provider class (MPC) based on the extension of the ABAP Dictionary include structure.

This method must be issued for each auto-expand include structure for an entity type and can be called before or after method `BIND_STRUCTURE`.

In addition, your MPC should also contain a reference of this ABAP Dictionary structure in form of data type declaration such as `LV_<DDIC_NAME> TYPE REF TO <DDIC_NAME>` to enable the framework to reload the model metadata once the ABAP Dictionary structure has been changed. The metadata will be immediately reloaded but the metadata cache on the SAP Gateway system will first be cleaned up at the end of the current request processing and the next request will automatically work with the new metadata.

Parameter	Description
IV_INCLUDE_NAME	Name of the ABAP Dictionary include structure.
IV_DUMMY_FIELD	This parameter is optional and defines a dummy field. That means it is not a property of the current entity type but it is necessary to enable the delivery of an "empty" ABAP Dictionary structure. This parameter is of type <code>FIELDNAME</code> .
IV_BIND_CONVERSIONS	This flag propagates information about conversion exits, currency and unit fields from ABAP Dictionary into metadata. The effect is the same as calling method <code>BIND_STRUCTURE</code> of interface <code>/IWBEP/IF_MGW_ODATA_ENTITY_TYP</code> . In addition, for the amount properties Precision and Scale are set according to the formula: Precision = Length * 2 - 1, Scale = Precision, where Length is an internal length of the structure field. This parameter is of type <code>ABAP_BOOL</code> with the default value <code>ABAP_FALSE</code> .

1.1.5.2.2.13 ABAP Dictionary Type to EDM.Type Mapping

A table showing how the metadata base class `/IWBEP/CL_MGW_ABS_MODEL` maps ABAP Dictionary types in case of structure binding to the EDM.Types

Use

The following table shows how the metadata base class `/IWBEP/CL_MGW_ABS_MODEL` maps ABAP Dictionary types in case of structure binding to the EDM.Types.

For the coding see `/IWBEP/CL_MGW_MED_EDM_UTIL->GET_EDM_TYPE_FROM_DDIC_TYPE`.

Edm. Type	Description	ABAP Type Kind	Description	Comment
Edm.Binary	Contains binary data.	TYPEKIND_HEX	Internal type X	
Edm.Binary	Contains binary data.	TYPEKIND_XSTRING	Internal type Y(byte string)	
Edm.Boolean	Contains the value true or false.	TYPEKIND_CHAR	Internal type C	If the number of characters is 1 and the name contains the substring <code>BOOL</code>
Edm.Byte	Contains an unsigned 8-bit integer value.	TYPEKIND_INT1	Internal type b (1 byte integer)	
Edm.DateTime	Represents a date and time.	TYPEKIND_DATE	Internal type D	
Edm.DateTime	Represents a date and time.	TYPEKIND_PACKED	P(8) if domain is <code>TZNTSTMP</code> , P(11) if domain is <code>TZNTSTMP_L</code> .	Edm.DateTime is set automatically if ABAP Dictionary domain is <code>TZNTSTMP</code> or <code>TZNTSTMP_L</code>
Edm.DateTimeOffset	Contains a date and time as an offset in minutes from GMT.	TYPEKIND_PACKED	Internal type P(8) or P(11) like for Edm.DateTime	Edm type of the property must be set explicitly in the metadata provider class.
Edm.Decimal	Contains a numeric value with fixed precision and scale.	TYPEKIND_PACKED	Internal type P	If the packed contains decimals
Edm.Decimal	Contains a numeric value with fixed precision and scale.	TYPEKIND_DECFLOAT		Only available in SAP NetWeaver 7.02, not SAP NetWeaver 7.00
Edm.Decimal	Contains a numeric value with fixed precision and scale.	TYPEKIND_DECFLOAT16		Only available in SAP NetWeaver 7.02, not SAP NetWeaver 7.00
Edm.Decimal	Contains a numeric value with fixed precision and scale.	TYPEKIND_DECFLOAT34		Only available in SAP NetWeaver 7.02, not SAP NetWeaver 7.00
Edm.Double	Contains a floating point number with 15 digit precision.	TYPEKIND_FLOAT	Internal type F	
Edm.Float	Contains a floating point number with seven digit precision.			
Edm.Single	Contains a floating point number with seven digit precision.			
Edm.Guid	Contains a 16-byte unique identifier.			Edm type of the property must be set explicitly in the metadata

				provider class to overrule the default value. Use <code>CHAR32</code> or <code>RAW16</code> in structure to support <code>Edm.Guid</code> and manually set the <code>Edm</code> type to <code>Edm.Guid</code> .
<code>Edm.Int16</code>	Contains a signed 16-bit integer value.	<code>TYPEKIND_INT2</code>	Internal type S(2 byte integer)	
<code>Edm.Int32</code>	Contains a signed 32-bit integer value.	<code>TYPEKIND_INT</code>	Internal type I	
<code>Edm.Int64</code>	Contains a signed 64-bit integer value.	<code>TYPEKIND_PACKED</code>	Internal type P	If the packed contains no decimals
<code>Edm.SByte</code>	Contains a signed 8-bit integer value.			
<code>Edm.String</code>	Contains character data.	<code>TYPEKIND_CHAR</code>	Internal type C	Number of characters is matched against the length. Unicode doubles the internal length.
<code>Edm.String</code>	Contains character data.	<code>TYPEKIND_STRING</code>	Internal type G (character string)	
<code>Edm.String</code>	Contains character data.	<code>TYPEKIND_NUM</code>	Internal type N	Input mask allowing only characters 0 - 9 is added
<code>Edm.Time</code>	Contains a time of day.	<code>TYPEKIND_TIME</code>	Internal type T	
		<code>TYPEKIND_W</code>	Internal type W(wide character)	For future evaluation
		<code>TYPEKIND_CLIKE</code>	Internal type (data object)	For future evaluation (proposal: <code>Edm.String</code>)
		<code>TYPEKIND_CSEQUENCE</code>	Internal type (data object)	For future evaluation (proposal: <code>Edm.String</code>)
		<code>TYPEKIND_XSEQUENCE</code>	Internal type (data object)	For future evaluation (proposal: <code>Edm.String</code>)
		<code>TYPEKIND_NUMERIC</code>	Internal type (data object)	For future evaluation (proposal: <code>Edm.String</code>)

1.1.5.2.2.14 ABAP Dictionary Type to EDM.Type Mapping SP10

For properties that are automatically added for each field of the so-called auto-expand include (simple field extensibility) there is a special EDM type determination logic in method `/IWBE/CL_MGW_MED_EDM_UTIL->GET_EDM_TYPE_V2`.

You can also use this EDM type determination for all non simple field extensibility properties instead of the default EDM determination by using method `/IWBE/CL_MGW_ODATA_MODEL->/IWBE/IF_MGW_ODATA_MODEL-SET_MODEL_FEATURES` and setting the model feature `use_edm_mapping_sp10`.

The following table shows how the metadata base class `/IWBE/CL_MGW_ABS_MODEL` maps ABAP Dictionary types in case this mapping is used.

Table 1:

Edm.Type	Description	ABAP Type Kind	Description	Comment
<code>Edm.Binary</code>	Contains binary data	<code>TYPEKIND_HEX</code>	Internal type X	
<code>Edm.Binary</code>	Contains binary data	<code>TYPEKIND_XSTRING</code>	Internal type Y (byte string)	
<code>Edm.Boolean</code>	Contains the value true or false	<code>TYPEKIND_CHAR</code>	Internal type C	<ul style="list-style-type: none"> If the number of characters is 1 and the domain contains the strings <code>BOOL</code> or <code>FLAG</code> If the domain is one of the following: <code>BOOLE</code>, <code>XFELD</code>, <code>XFLAG</code>, <code>FLAG</code>, <code>X</code>, <code>DDFLAG</code>, <code>CHAR1_X</code>
<code>Edm.Byte</code>	Contains an unsigned 8-bit integer value	<code>TYPEKIND_INT1</code>	Internal type B (1 byte integer)	
<code>Edm.DateTime</code>	Represents a date and time	<code>TYPEKIND_DATE</code>	Internal type D	
<code>Edm.DateTimeOffset</code>	Represents a date and time	<code>TYPEKIND_PACKED</code>	P(8) if domain is <code>TZNTSTMP</code> , P(11) if domain is <code>TZNTSTMP1</code>	If the ABAP Dictionary domain is <code>TZNTSTMP</code> or <code>TZNTSTMP1</code>
<code>Edm.Decimal</code>	Contains a numeric value with fixed precision and scale	<code>TYPEKIND_PACKED</code>	Internal type P	If ABAP Dictionary domain is not <code>TZNTSTMP</code> or <code>TZNTSTMP1</code>
<code>Edm.Decimal</code>	Contains a numeric value with fixed precision and scale	<code>TYPEKIND_DECFLOAT</code>		Only available in systems based on SAP NetWeaver 7.02, not 7.00
<code>Edm.Decimal</code>	Contains a numeric value with fixed precision and scale	<code>TYPEKIND_DECFLOAT16</code>		Only available in systems based on SAP NetWeaver 7.02, not 7.00
<code>Edm.Decimal</code>	Contains a numeric value with fixed precision and scale	<code>TYPEKIND_DECFLOAT34</code>		Only available in systems based on SAP NetWeaver 7.02, not

				7.00
Edm.Double	Contains a floating point number with 15 digit precision	Internal type F	Internal type F	
Edm.Guid	Contains a 16-byte unique identifier	TYPEKIND_HEX		<ul style="list-style-type: none"> • If ABAP Dictionary domain is SYSUUID • If the length is 16 and the domain name contains the strings UUID or GUID • If the length is 16 and the element name is GUID
Edm.Guid	Contains a 16-byte unique identifier	TYPEKIND_CHAR		If ABAP Dictionary domain is SYSUUID_22, SYSUUID_C22 or SYSUUID_C.
Edm.Int16	Contains a signed 16-bit integer value	TYPEKIND_INT2	Internal type S (2 byte integer)	
Edm.Int32	Contains a signed 32-bit integer value	TYPEKIND_INT	Internal type I	
Edm.String	Contains character data	TYPEKIND_CHAR	Internal type C	Only if Edm.Boolean or Edm.Guid was not determined (see related comments)
Edm.String	Contains character data	TYPEKIND_STRING	Internal type G (character string)	
Edm.String	Contains character data	TYPEKIND_NUM	Internal type N	Input mask allowing only characters '0'-'9' is added
Edm.Time	Contains a time of day	TYPEKIND_TIME	Internal type T	

Related Information

[/IWBEP/IF_MGW_ODATA_MODEL](#)

1.1.5.2.2.15 /IWBEP/IF_MGW_ODATA_ITEM

Use

This interface is implemented by every OData Channel type.

Methods

Method SET_NAME

This method sets the name of the OData Channel type.

Parameter	Description
IV_NAME	Value

Method GET_ID

This method gets the ID of the OData Channel type, that is, it returns the ID of the OData Channel entity.

Parameter	Description
RV_ID	Value

Method SET_LABEL_FROM_TEXT_ELEMENT

This method defines the <sap:label> of an OData artifact in the service metadata document.

Parameter	Description
IV_TEXT_ELEMENT_SYMBOL	Symbol of the text element, for example, 000.
IV_TEXT_ELEMENT_CONTAINER	Name of the text element container, for example, either a report name or the class name. This is an optional parameter which should be filled if the text reference is located in another class or report and not the MPC.
IO_OBJECT_REF	Reference of the class which contains the text element. If the MPC contains the texts ABAP me needs to be passed.

Method CREATE_DOCUMENTATION

This method creates a documentation object for an artifact.

Parameter	Description
ro_documentation	type ref to /iwbsp/if_mgw_odata_document

1.1.5.2.2.16 /IWBEP/IF_MGW_ODATA_MODEL

Use

This interface is the entry point for every OData Channel model definition to define the required OData artifacts.

Age HTTP Header

The data provider may set an age in the data provider methods for `GET_ENTITY_TYPE` / `GET_ENTITY_SET` which is translated into an HTTP response header without further calculation.

Methods

Method CREATE_ACTION

This method adds an action, service operation, or function import to the OData model.

Parameter	Description
IV_ACTION_NAME	Value
RO_ACTION	Instance

Method CREATE_TAG

This method creates a service tag: an action or an OData function import. Actions are pre-packaged chunks of business logic that go beyond simple CRUD operations on data object instances.

Table 1:

Parameter	Description
IV_ACTION_NAME	Importing the external name of the action as it will be available in the service document
RO_ACTION	Returns the created action

For more information, see [Service Tagging](#).

Method GET_ACTION

This method retrieves an action from a OData model by name.

Parameter	Description
IV_ACTION_NAME	Value
RO_ACTION	Instance

Method GET_COMPLEX_TYPE

This method gets a complex type by name.

Parameter	Description
IV_CPLX_TYPE_NAME	Value
RO_COMPLEX_TYPE	Instance

Method CREATE_COMPLEX_TYPE

This method adds a complex type to the OData model.

Parameter	Description
IV_CPLX_TYPE_NAME	Value
RO_COMPLEX_TYPE	Instance

Method CREATE_ENTITY_TYPE

This method adds an EntityType to the OData model.

Parameter	Description
IV_ENTITY_TYPE_NAME	Value
RO_ENTITY	Instance

Method GET_ENTITY_TYPE

This method gets an EntityType by name.

Parameter	Description
IV_ENTITY_TYPE_NAME	Value
RO_ENTITY	Instance

Method CREATE_ASSOCIATION

This method adds an association to the data model which links to EntityTypes.

Parameter	Description
IV_ASSOCIATION_NAME	Value
IV_LEFT_TYPE	Name of the left EntityType.
IV_RIGHT_TYPE	Name of the right EntityType.
IV_LEFT_CARD	Association cardinality of the left EntityType.
IV_RIGHT_CARD	Association cardinality of the right EntityType.
RO_ASSOCIATION	Instance

Method GET_ASSOCIATION

This method gets an association of the data model.

Parameter	Description
IV_ASSOCIATION_NAME	Value
RO_ASSOCIATION	Instance

Method SET_NO_CONVERSION

This method controls whether any conversion should be called for runtime representations of this data model.

Parameter	Description
IV_NO_CONVERSION	Flag to decide whether any conversion should be called during runtime.

Method GET_NO_CONVERSION

This method indicates whether conversions are called or not, during runtime.

Parameter	Description
RV_NO_CONVERSION	Value

Method INCLUDE_MODEL_BY_NAME

This method includes a different data model by name from another OData Channel service exposed on SAP Gateway, that is, `IW_BEP` data models. The API can be used to compose data models from different services, that is, to include an external data model. The composition is defined in the system in which the software component `IW_BEP` is installed. The software component `IW_BEP` can be installed on an SAP Business Suite backend system or on an SAP Gateway hub system.

Note

If your system is based on SAP NetWeaver 7.4 you do not need to install component `IW_BEP` since the SAP Gateway Foundation component `SAP_GWFND` is installed as standard.

The service of the external data model must be registered on the SAP Gateway hub system and be up and running.

The ID of the service and the name of the data model have to be provided to the method. This information is stored on the SAP Gateway hub and is resolved during runtime to create, for example, the metadata document and the business data for the routing. This means that for entities of the external data model, the system alias assigned to the service of the external data model is used instead of the system alias assigned to the original service including that data model.

During activation, the service maintenance function uses the provided service IDs and data model names to find a corresponding service. Therefore the included services have to be activated first, as otherwise the activation of the composition service will lead to an incomplete composition service or even an error.

It is possible to define associations to the entities of the included data model, although they do not exist at this time. 1:1 and 1:n relations can be defined pointing from the new data model to the included external data model.

- When defining the association, you define the left side of it (from the current data model existing in `IW_BEP`) with the standard method `/IWBEF/IF_MGW_ODATA_MODEL~CREATE_ASSOCIATION`.
- It is necessary to define referential constraints for these associations as they are resolved generically during runtime by the framework on the SAP Gateway hub system.

During the runtime based on these referential constraints a request like `Flight(ID=100)/BookingsListedInHana` might be converted either into a request `BookingsListedInHana(ID=100)` or `BookingsListedInHana?$filter=ID eq '100'` depending on the referential constraints. If the keys of the target entity fit either to the key of the source entity or are mapped appropriately in the referential constraints then the key notation is used, otherwise the filter is filled. The resolution behaviour can be overruled by your own BAdI implementation (see below).

Note

It is currently **not** possible to define navigation properties for entities of included data models.

Enhancement Spot `/IWFND/ES_MGW_MDL_COMPOSITION` can be used to overwrite the link resolution behavior.

You can find an example in method `DEFINE` of class `/IWBEF/CL_MGW_MED_EXT_COMP_MDL`.

Parameter	Description
IV_TECH_MODEL_NAME	Technical name of model.

IV_MODEL_VERSION	Technical version of model.
IV_EXT_SERVICE_NAME (Optional)	External name of the service as specified in the BEP registration (optional).
IV_SERVICE_VERSION (Optional)	Technical version of the service (optional).

Details

For every association, an association set is required. The creation of the association set is done automatically when calling method `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION`.

In case of model composition, the creation of the association set is not automatically. Instead, do the following:

- Deactivate the defaulting by setting the import parameter `IV_DEF_ASSOC_SET` to `FALSE` in method `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION`.
- Create the corresponding association set explicitly by calling `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION_SET`

Caution

The model include is done dynamically. It does not need a registration of the depending service or model during the registration of the hosting service. Changes are directly reflected after the metadata cache is invalidated.

Method `EXTEND_MODEL`

This method extends an ODC model using the model name / model version (model ID). This method overwrites all existing defined metadata by the metadata of the extended model. So use this method only once and as the first statement:

This method overwrites all existing defined metadata by the metadata of the extended model. So use it as the first statement and only once in the model definition. Multiple extensions are not supported. `Extend_Model` only supports the metadata extension of a model, and has no impact on the run time behavior.

Note

In the model definition, multiple extensions are not supported. The `Extend_Model` method only supports the metadata extension of a model and has no impact on the runtime behavior.

Parameter	Description
IV_TECH_MODEL_NAME	Technical name of model
IV_MODEL_VERSION	Technical version of model

`SET_MODEL_FEATURES`

This method sets different model features via parameter `IS_MODEL_FEATURES`. The following model features can be set at present:

- `use_cache_handshake_busi_req` (activates the cache handshake for business requests)
- `use_edm_mapping_sp10` (use the EDM mapping as developed in support package 10)
Note that the default is the previous EDM mapping.
- `use_long_label_for_property`

Caution

This method should be placed in the coding at the beginning of the `DEFINE` method of your model provider class (MPC) to make sure that the model features will be valid for all model artifacts.

Cache Handshake for Business Requests

By setting this model feature the cache handshake for OData business requests will be activated for the service. With OData business requests all read/create/update/delete requests are meant, but not \$metadata or service document requests. The cache handshake for business requests makes sure that an exception will be raised and the response will have the HTTP status code 503 (service unavailable) in case an OData business request was executed and the metadata cache on your hub system was outdated. After this error had occurred, the metadata cache on your hub system should be cleaned-up automatically by the system. The next OData business request of this service should work. Therefore, simply execute the OData business request again that has led to this error (or in fact any other OData business request of this service). If the automatic cache clean-up did not work, you need to clean up the metadata cache manually on your hub system for the current model.

EDM Mapping in Support Package 10

Use EDM mapping from support package 10 and note that the default mode is the previous EDM mapping.

Use Long Label for Property

Labels for properties based on ABAP Dictionary elements will be taken as default from the long description if the long text does not exceed 20 characters.

Code Syntax

Example for enabling the cache handshake for business requests

```
METHOD define.

    DATA ls_model_features TYPE /iwbsp/if_mgw_appl_types=>ty_s_model_features.

    super->define( ).

    ls_model_features-use_cache_handshake_busi_req = abap_true.
```

```
model->set_model_features( ls_model_features ).
```

```
ENDMETHOD.
```

Method SET_SERVICE_SCHEMA_VERSION

The application can use this method to set the SAP schema version, for example `<Schema Namespace="com.sap.TEA.TestApplication" xml:lang="en" sap:schema-version="0001" xmlns="http://schemas.microsoft.com/ado/2008/09/edm">`.

This can be considered a sub-version of the service version. If not set explicitly, the default schema version is 0000. This is not supported for service composition. That is, in case of service composition the value is always 0000.

Table 2:

Parameter	Description
IV_SERVICE_SCHEMA_VERSION	Schema version

Method SET_SCHEMA_NAMESPACE

An application developer can use this method to set the schema namespace attribute for the metadata document.

Table 3:

Parameter	Description
IV_NAMESPACE	Namespace

The schema namespace appears in the metadata document as follows:

Code Syntax

```
<Schema Namespace="Tea_test_application_NS" xml:lang="en"..
<Property Name="Location" Type="Tea_test_application_NS.CT_Location" Nullable="false" />
<NavigationProperty Name="My_Team" Relationship="Tea_test_application_NS.Team_of_Employee" FromRole="FromRole_Te
|||
<Association Name="Team_of_Employee" sap:content-version="1">
<End Type="Tea_test_application_NS.Worker" Multiplicity="*" Role="FromRole_Team_of_Employee" />
<End Type="Tea_test_application_NS.Team" Multiplicity="1" Role="ToRole_Team_of_Employee" />
<EntityContainer Name="Tea_test_application_NS.Entities" m:IsDefaultEntityContainer="true">
<ComplexType Name="CT_Location">
<Property Name="Country" Type="Edm.String" MaxLength="255" sap:text="Country" />
<Property Name="City" Type="Tea_test_application_NS.CT_City" Nullable="false" />
<Property Name="Complex_type_City" Type="Tea_test_application_NS.CT_City" Nullable="false" />
</ComplexType>
```

Method INCLUDE_MODEL_BY_SERVICE_ID

Note

This method is obsolete and should not be used any more.

This method Includes a model from another OData Channel Service exposed on the SAP Gateway system, that is backend models. The API can be used to compose Object Models from different services, that is to include an external model. For example, the composition is defined in the backend system system, but resolved on the SAP Gateway system.

To attain this, the following conditions should be met:

- The service of the external model must exist on the SAP Gateway system and be up and running.
- The ID of the service and the name of the model have to be provided to the method. This information is stored on the SAP Gateway and resolved during runtime to create the metadata document and for the business data for the routing. This means that for entities of the external model the system alias assigned to the service of the external model is used instead of the system alias assigned to the original service including that model.
- During activation the **Registration Report** uses the provided service IDs and model names to find a corresponding service. Therefore the included services should be activated first, otherwise the activation of the composition service will lead to an incomplete composition service or might even throw errors.

It is possible to define associations to the entities of the included model, even if they do not exist at that time. 1:1 and 1:n relations can be defined pointing from the new model to the included external model.

- When defining the association you define the left side of it (from the current model existing in BEP) with the standard method `/IWBEP/IF_MGW_ODATA_MODEL~create_association`
- It is necessary to define referential constraints for these associations as they are resolved generically during runtime by the framework on the SAP Gateway.

During runtime based on these referential constraints a request like `Flight(ID=100)/BookingsListedInHana` is converted into a request `BookingsListedInHana?$filter=ID eq '100` to the HANA data provider.

Note

Currently, it is not possible to define navigation properties for entities of included models. In future, an exit user (BADI) is planned to overwrite the link resolution (and possibly also the dispatching) behavior.

Example

Method DEFINE of class `/IWBEP/CL_MGW_MED_EXT_COMP_MDL`

Details

For every association an association set is required. The creation of the association set is done automatically when calling the method:

- `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION`

In case of model composition, the creation of the association set is not automatic. Instead, do the following:

- Deactivate the defaulting by setting the import parameter `IV_DEF_ASSOC_SET` to `FALSE` in method `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION`.
- Create the corresponding association set explicitly by calling `/IWBEP/IF_MGW_ODATA_MODEL->CREATE_ASSOCIATION_SET`

Note

The composition must have been done BEFORE the service is created on the SAP Gateway system via transaction `/IWFND/MAINT_SERVICE`.

Method INCLUDE_MODEL_BY_MAPPING_ID

Note

This method is obsolete and should not be used any more.

This method includes a model from generic SAP Gateway Integration Service exposed on the SAP Gateway, for example, . HANA or BW services. The API can be used to compose Object Models from different services, that is to include an external model. For example, the composition is defined in the backend system, but resolved on the SAP Gateway system.

To attain this, the following conditions should be met:

- The service of the external model must exist on the SAP Gateway system and be up and running.
- The External Mapping ID of the service as displayed on the SAP Gateway has to be provided to the method interface. This information is stored on the SAP Gateway and resolved during runtime to create for example, the metadata document and for the business data for the routing. This means that for entities of the external model, the system alias assigned to the service of the external model is used instead of the system alias assigned to the original service including that model.
- During activation the **Registration Report** uses the External Mapping ID to find a corresponding service. Therefore the included services should be activated first, otherwise the activation of the composition service will lead to an incomplete composition service or might even throw errors.

It is possible to define associations to the entities of the included model, although they do not exist at this time. `1:1` and `1:n` relations can be defined pointing from the new model to the included external model.

- When defining the association you define the left side of it (from the current model existing in BEP) with the standard method `/IWBEP/IF_MGW_ODATA_MODEL~create_association`
- It is necessary to define referential constraints for these associations as they are resolved generically during runtime by the framework on the SAP Gateway.

During runtime based on these referential constraints, a request like **Flight(ID=100)/BookingsListedInHana** is converted into a request **BookingsListedInHana?\$filter=ID eq '100** to the HANA data provider.

In the future, an exit user (BADL) is planned to overwrite the link resolution (and possibly also the dispatching) behavior.



Example

Method DEFINE of class `/IWBEP/CL_MGW_MED_BW_COMP_MDL`

1.1.5.2.2.17 /IWBEP/IF_MGW_ODATA_NAV_PROP

Use

This interface represents an OData navigation property object.

Method SET_TITLE

Note

This method is **deprecated**. Use method [SET_LABEL_FOR_TEXT_ELEMENT](#) instead.

This method sets the title of the navigation property.

This method thus sets the title for `atom:link` in an entry which points to the referenced entity. A source of this text can be a text symbol of a class.

The semantics of the input parameter must be `CLASS NAME` and the symbol Id of the text element separated by `-`. The corresponding text object key is defined via `/IWBEP/CL_MGW_ABS_MODEL`.

Parameter	Description
<code>IV_TEXT_KEY</code>	Text key in the class name and the text element separated by <code>-</code> . Example: <code>/IWBEP/CL_MGW_MED_POOL_TX_LOAD-000</code> (case insensitive).
<code>IV_TEXT_OBJ_TYPE</code>	Object type in the text object type defined in the field mentioned above. Example <code>GCS_SAP_TEXT_OBJECT_TYPES-BACKEND_GATEWAY_TEXT</code> .

1.1.5.2.2.18 /IWBEP/IF_MGW_ODATA_PARAMETER

Use

1.1.5.2.2.19 /IWBEP/IF_MGW_ODATA_PROPERTY

Use

This is an interface which represents an OData Channel property object.

Methods

Method SET_TYPE_EDM_BINARY

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_BYTE

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_DATETIME

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_DATETIMEOFFSET

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_DECIMAL

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_DOUBLE

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_FLOAT

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_SINGLE

This method sets the EDM Core type of property to SINGLE. The type EDM.Single is handled in the same way as the type EDM.Float at runtime.

Example:

```
<EntityType sap:content-version="*1*">
  <Key>
    <PropertyRef />
    <PropertyRef />
  </Key>
  <Property Type="*Edm.String*" Nullable="*false*" MaxLength="*4*" />
  ...
  <Property Type="*Edm.Single*" />
</EntityType>
```

Method SET_TYPE_EDM_GUID

This method sets the EDM core type of a property.

Method Method SET_TYPE_EDM_INT16

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_INT32

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_INT64

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_SBYTE

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_STRING

This method sets the EDM core type of a property.

Method SET_TYPE_EDM_TIME

This method sets the EDM core type of a property.

Method SET_FIELD_CONTROL

This method sets the property that contains the field control, it creates the annotation `sap:field-code="SomeOtherProperty"`.

Method SET_PRECISION

This method sets the precision of a property. Note that this is only applicable for certain EDM core types.

Parameter	Description
IV_PRECISION	Precision of the property.

Method SET_MAXLENGTH

This method sets the maximal length of a property, for example for an EDM.string. Note that this is only applicable for certain EDM core types.

Parameter	Description
IV_MAX_LENGTH	Maximum length of the property.

Method SET_CONVERSION_EXIT

This method sets the name of conversion exit (for example, ALPHA) to be executed for the property in in-/outbound flow.

If a conversion exit has been set, the corresponding functions (for example, `CONVERSION_EXIT_ALPHA_INPUT` and `CONVERSION_EXIT_ALPHA_OUTPUT`) are executed by the framework automatically during inbound and outbound flow.

Parameter	Description
IV_CONV_EXIT	Name of the conversion exit. /IWBEF/IF_MGW_MED_ODATA_TYPES=>TY_E_MED_CONV_EXIT

Method SET_NULLABLE

This method sets the Nullable attribute for a property in the service metadata document, for example, `Nullable=true` or `Nullable=false`.

Parameter	Description
IV_NULLABLE	Value of the property.

Method SET_CREATABLE

This method sets the value of `sap:creatable` of the property. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_CREATABLE	Value of the property.

Method SET_UPDATABLE

This method sets the value of `sap:updatable` of the property. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_UPDATABLE	Value of the property.

Method SET_FILTERABLE

This method sets the value of `sap:filterable` of the property. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_FILTERABLE	Value of the property.

Method SET_SORTABLE

This method sets the value of `sap:sortable` of the property. The default value is `true` and therefore not shown in service metadata document.

Parameter	Description
IV_SORTABLE	Value of the property.

Method SET_IS_KEY

This method sets the property as key of the enclosing entity type.

Parameter	Description
IV_KEY	Value of the property.

Method SET_SEMANTIC

This method sets the semantics for the property. Supported values are the components of the structure `/IWBEF/CL_MGW_ABS_MODEL=>GCS_SAP_SEMANTIC`, in particular the following:

- CL_MGW_ABS_MODEL=>GCS_SAP_SEMANTIC-CURRENCY_CODE
Currency code. If set, conversion between SAP internal code (for example, DEM3) and external (ISO) one (for example, DEM) is performed automatically by the framework for inbound and outbound flow.
- CL_MGW_ABS_MODEL=>GCS_SAP_SEMANTIC-UNIT_OF_MEASURE
Unit of measurement. If set, conversion between SAP internal code (for example, KG) and external (ISO) one (for example, KGM) is performed automatically by the framework for inbound and outbound flow.

Parameter	Description
IV_SEMANTIC	Semantics. /IWBEF/IF_MGW_MED_ODATA_TYPES=>TY_E_MED_SEMANTIC

Method SET_INTERNAL_TYPE

This method sets the internal (ABAP) type of a property.

Parameter	Description
IV_TYPE	Type based on the ABAP type system of the property.

Method SET_INTERNAL_LENGTH

This method sets the internal (ABAP) length of a property. Usage in exceptional cases only.

Sets the internal ABAP length of the property. This information is usually derived from the underlying DDIC binding. The method should only be used in exceptional cases. The method is purely a setter method and doesn't do any calculation. This has to be done up-front and all circumstances, for example Unicode or non-Unicode have to be taken into account.

Parameter	Description
IV_INTERNAL_LENGTH	Internal ABAP length of property.

Method SET_TEXT_KEY

This method sets the source of a text label, for example, a text element or ABAP Dictionary label.

The method thus creates a description for a field that differs from those of the ABAP Dictionary. A source of this text can be a text symbol of a class. The semantics of the input parameter must be the `CLASS NAME` and the symbol Id of the text element separated by -. The corresponding text object key is defined in class `/IWBEP/CL_MGW_ABS_MODEL`.

Parameter	Description
IV_TEXT_KEY	Text key in the class name and the text element separated by -. Example: <code>/IWBEP/CL_MGW_MED_POOL_TX_LOAD-000</code>
IV_TEXT_OBJECT_TYPE	Text object type defined in the field mentioned above. Example: <code>GCS_SAP_TEXT_OBJECT_TYPES-BACKEND_GATEWAY_TEXT</code>
IV_CREATE	In the text the key will be created if not available. This parameter must always be set to <code>ABAP_TRUE</code> .

Method BIND_DATA_ELEMENT

This method binds a property to given data element.

If a data element has been bound to a property, type (ABAP and EDM core) and label information are retrieved from the ABAP Dictionary.

Parameter	Description
IV_ELEMENT_NAME	Name of the data element in the ABAP Dictionary.

Method BIND_DATA_ELEMENT_FOR_TEXT

Entity types bound to ABAP Dictionary structures might want to exchange bindings of single properties to data elements. For this method `BIND_DATA_ELEMENT_FOR_TEXT` of interface `/IWBEP/IF_MGW_ODATA_PROPERTY` is provided. With this method you can overwrite the text source for a property, that is, it binds a property to given data element for text. This method declares the given data element from the ABAP Dictionary as text source for this property.

Parameter	Description
IV_ELEMENT_NAME	Name of the data element in the ABAP Dictionary.

Method SET_UNIT

Obsolete. Use method `SET_UNIT_PROPERTY` instead.

Method SET_UNIT_PROPERTY

This method declares another property as measurement unit for the current one.

If the property set as unit for the current one has currency semantic (`/cl_mgw_abs_model=>gcs_sap_semantic-currency_code`) the current property is considered to be a currency amount property. This causes currency amount conversion to be executed automatically by the framework during inbound and outbound flow. This is normally required for every currency amount field since the internal (DB) representation may be differ from the external (real) one, for example `10.000 JPY` are stored internally as `100.00`.

Parameter	Description
IV_UNIT_PROPERTY	Type <code>/IWBEP/IF_MGW_MED_ODATA_TYPES=>TY_E_MED_ENTITY_NAME</code> - Name of the property, set as measurement unit for the current one.

Method SET_PRECISION

This method is obsolete. Use method `SET_UNIT_PRECISION_PROPERTY` instead.

Method SET_UNIT_PRECISION_PROPERTY

This method sets the property that contains the precision for the current unit property.

Parameter	Description
IV_PRECISION_PROPERTY	Type <code>/IWBEP/IF_MGW_MED_ODATA_TYPES=>TY_E_MED_ENTITY_NAME</code> - Name of the property set as precision for the current one

Method /IWBEP/IF_MGW_ODATA_PROPERTY~SET_REF_ANNO_PROPERTY

This method sets the property that contains the precision for the current unit property.

The method allows to set all annotation keys generically. Currently, there is a check on the backend system for the following annotation keys:

- `sap:text`
- `sap:unit`
- `sap:precision`
- `sap:lower-boundary`
- `sap:upper-boundary`
- `sap:super-ordinate`

- sap:attribute-for
- sap:hierarchy-node-for
- sap:hierarchy-level-for
- sap:hierarchy-parent-node-for
- sap:hierarchy-drill-state-for

Parameter	Description
IV_ANNOTATION_KEY	Name of the annotation element, of type /IWBEP/MED_ANNOTATION_KEY.
IV_PROPERTY	Name of the property referred to in the annotation, of type /IWBEP/MED_EXTERNAL_NAME.

On the SAP Gateway hub system the GUID of the referenced property is replaced by the actual name of the property. This method can be used instead of SET_UNIT_PROPERTY and SET_UNIT_PRECISION_PROPERTY.

Method SET_FC_TARGET_PATH

This method sets the target path of feed customization to property.

The SET_AS* methods represent concrete target paths and wrap the SET_FC_TARGET_PATH method.

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method SET_AS_AUTHOR

This method sets the target path of the property to feed customization target SyndicationTitle (atom:author).

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method SET_AS_ETAG

This method sets the target path of the property to the ETag of the entry (m:etag in <atom:entry>) and also to the HTTP response header of a single read. If the **Compatibility Mode for SP 02** is used the ETag needs to be set manually in the DPC. If the **Standard Mode** is used the ETag is set automatically. See also <http://www.odata.org/developers/protocols/operations#ConcurrencycontrolandETags> 📄.

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method SET_AS_TITLE

This method sets the target path of the property to feed the customization target SyndicationTitle (atom:title).

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method SET_AS_UPDATED

This method sets the target path of the property to feed the customization target SyndicationUpdated (atom:updated).

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method SET_AS_PUBLISHED

This method sets the target path of the property to feed the customization target SyndicationPublished(atom:published).

Parameter	Description
IV_KEEP_IN_CONTENT	Boolean to decide whether to keep the property still in the content or only in the target of the property.

Method DISABLE_CONVERSION

This method disables the conversion for the property. For more information, see [1893788](#) 📄.

Method ENABLE_CONVERSION

This method enables the conversion for the property. For more information, see [1893788](#) 📄.

1.1.5.2.2.20 /IWBEP/IF_MGW_ODATA_REF_CONSTR

Use

This interface represents a referential constraint construct in OData and is linked to an association.

Method ADD_PROPERTY

This method adds a link between the primary and foreign key property of the principal entity type and dependent entity type.

Parameter	Description
IV_PRINCIPAL_PROPERTY	Name of the primary key property in the principal entity type (name and not the ABAP name).
IV_DEPENDENT_PROPERTY	Name of the foreign key property in the dependent entity type (name and not the ABAP name).

1.1.5.2.3 OData Vocabulary Annotations APIs

Overview of vocabulary annotations and their interfaces and methods.

Use vocabulary annotations to extend your OData services. Annotating an OData service enables you to add information that tells the clients of your service how they should interpret the service and its data. You can use annotations to add semantics that describe certain behavior that is applicable to a particular artifact, such as creatable, updatable, or sortable, for example. Annotations also enable you to provide further information such as which fields form an address or which fields logically belong together.

For example, if you have an OData service that is integrated with a mobile device to provide contact details and phone numbers, you can use annotations to tell the client which of the OData properties in this service contain the contact's details such as name and address and which properties contain a phone number.

Vocabulary Annotations Interfaces and Methods

The following interfaces and methods are available for annotating OData services :

- [/IWBEP/IF_MGW_VOCAN_ANNOTATION](#)
- [/IWBEP/IF_MGW_VOCAN_ANN_TARGET](#)
- [/IWBEP/IF_MGW_VOCAN_COLLECTION](#)
- [/IWBEP/IF_MGW_VOCAN_FUNCTION](#)
- [/IWBEP/IF_MGW_VOCAN_FUN_PARAM](#)
- [/IWBEP/IF_MGW_VOCAN_LABEL_ELEM](#)
- [/IWBEP/IF_MGW_VOCAN_PROPERTY](#)
- [/IWBEP/IF_MGW_VOCAN_RECORD](#)
- [/IWBEP/IF_MGW_VOCAN_SIMPLE_VAL](#)
- [/IWBEP/IF_MGW_VOCAN_TERM](#)
- [/IWBEP/IF_MGW_VOCAN_URL](#)

Support of Conditional Expressions


You can use the capabilities of dynamic expressions for UI field control. The following expressions are available: IF , =, <>, >, >=, <, <=, AND, OR, NOT

 **Sample Code**

```
<Property Name="Reiseland" Type="Edm.String">
  <Annotation Term="Common.FieldControl">
    <If>
      <Eq>
        <Path>Auslandsreise</Path>
        <Bool>true</Bool>
      </Eq>
      <Path>EditState</Path>
      <Path>Active</Path>
    </If>
  </Annotation>
</Property>
```

Different subtypes:

- IF: Condition, then, else
- Not: One child
- Other: Two children
- Works with inline and annotation file

 **Sample Code**

```
/ iwbeep/if_mgw_vocan_cond_exp = /iwbeep/if_mgw_vocan_annotation->create_con_exp( /iwbeep/if_mgw_med_odata_types=>
/ iwbeep/if_mgw_vocan_cond_exp_p = / iwbeep/if_mgw_vocan_cond_exp-> create_parameter( )
/ iwbeep/if_mgw_vocan_cond_exp_p->
CREATE_SIMPLE_VALUE()
CREATE_CON_EXPRESSION()
CREATE_RECORD()
CREATE_FUNCTION()
CREATE_LABELED_ELEMENT()
```

1.1.5.2.3.1 /IWBEP/IF_MGW_VOCAN_ANNOTATION

OData vocabulary annotation interface and methods.

You can create annotations from an annotation target or from a term. However, labeled elements, properties, records, and even annotations themselves can also contain embedded annotations.

Example

In the following example, an annotation has been created from an annotation target:

DATA:

```
lo_ann_target TYPE REF TO /iwbsp/if_mgw_vocan_ann_target, " Vocabulary Annotation Target

lo_annotation TYPE REF TO /iwbsp/if_mgw_vocan_annotation. " Vocabulary Annotation


lo_ann_target = vocab_anno_model->create_annotations_target( 'Products' ).

lo_annotation = lo_ann_target->create_annotation( iv_term = 'Org.OData.Core.V1.OptimisticConcurrencyControl' ).
```

In the metadata document this is reflected as follows:

```
<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="Products">

  <Annotation Term="Org.OData.Core.V1.OptimisticConcurrencyControl"/>

</Annotations>
```

Methods

Table 1: Methods Available for /IWBEP/IF_MGW_VOCAN_ANNOTATION

Name of Method	Function
CREATE_ANNOTATION	Creates an annotation within an annotation
CREATE_COLLECTION	Creates a collection
CREATE_RECORD	Creates a record
CREATE_FUNCTION	Creates a function
CREATE_LABELED_ELEMENT	Creates a labeled element
CREATE_SIMPLE_VALUE	Creates a simple value
CREATE_URL	Creates a URL

1.1.5.2.3.2 /IWBEP/IF_MGW_VOCAN_ANN_TARGET

OData vocabulary annotation interface and methods.

See how annotation target is used in context in [/IWBEP/IF_MGW_VOCAN_ANNOTATION](#).

Methods

Table 1: Methods available for /IWBEP/IF_MGW_VOCAN_ANN_TARGET

Method	Function
CREATE_ANNOTATION	Creates a vocabulary annotation
SET_NAMESPACE_QUALIFIER	Sets an alias prefix to qualify the target

1.1.5.2.3.3 /IWBEP/IF_MGW_VOCAN_COLLECTION

OData vocabulary annotation interface and methods.

Annotations, labeled elements, and properties can contain collections.

A collection can contain records and simple values. In the code example a collection is created for an annotation. The collection contains a simple value (string):

Example

DATA:

```
lo_ann_target TYPE REF TO /iwbsp/if_mgw_vocan_ann_target, " Vocabulary Annotation Target
```

```

lo_annotation TYPE REF TO /iwbsp/if_mgw_vocan_annotation, " Vocabulary Annotation

lo_simp_value TYPE REF TO /iwbsp/if_mgw_vocan_simple_val, " Vocabulary Annotation Simple Value

lo_collection TYPE REF TO /iwbsp/if_mgw_vocan_collection. " Vocabulary Annotation Collection


lo_ann_target = vocab_anno_model->create_annotations_target( 'Products' ).

lo_annotation = lo_ann_target->create_annotation( iv_term = 'Org.OData.Core.V1.OptimisticConcurrencyControl' ).

lo_collection = lo_annotation->create_collection( ).

lo_simp_value = lo_collection->create_simple_value( ).

lo_simp_value->set_string( 'image/jpeg' ).

```

In the metadata document, this is reflected as follows:

```

<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="Products">

  <Annotation Term="Org.OData.Core.V1.OptimisticConcurrencyControl">

    <Collection>

      <String>image/jpeg</String>

    </Collection>

  </Annotation>

</Annotations>

```

Methods

Table 1: Methods available for /IWBEF/IF_MGW_VOCAN_COLLECTION

Method	Function
CREATE_RECORD	Creates a record
CREATE_SIMPLE_VALUE	Creates a simple value

1.1.5.2.3.4 /IWBEF/IF_MGW_VOCAN_FUNCTION

OData vocabulary annotation interface and methods.

Interface /IWBEF/IF_MGW_VOCAN_FUNCTION allows the definition of functions. By using method CREATE_PARAMETER, you can add parameters to your function, see code example.

Example

Use method CREATE_PARAMETER to add parameters to your function. For example:

```

DATA:

lo_ann_target TYPE REF TO /iwbsp/if_mgw_vocan_ann_target, " Vocabulary Annotation Target

lo_annotation TYPE REF TO /iwbsp/if_mgw_vocan_annotation, " Vocabulary Annotation

lo_function TYPE REF TO /iwbsp/if_mgw_vocan_function, " Vocabulary Annotation Function

lo_fun_param TYPE REF TO /iwbsp/if_mgw_vocan_fun_param, " Vocabulary Annotation Function Parameter

lo_simp_value TYPE REF TO /iwbsp/if_mgw_vocan_simple_val. " Vocabulary Annotation Simple Value


lo_ann_target = vocab_anno_model->create_annotations_target( 'My Anno Target' ).

lo_annotation = lo_ann_target->create_annotation( iv_term = 'Core.AcceptableMediaTypes'

                                                    iv_qualifier = 'iPad' ).

lo_function = lo_annotation->create_function( 'odata.concat' ).

lo_fun_param = lo_function->create_parameter( ).

lo_simp_value = lo_fun_param->create_simple_value( ).

```

```
lo_simp_value->set_path( 'Name' ).
```

In the metadata document, this is reflected as follows:

```
<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="My Anno Target"><Annotation Qualifier="iPad"

    <Apply Function="odata.concat">

        <Path>Name</Path>

    </Apply></Annotation>

</Annotations>
```

Method

Table 1: Method for /IWBEP/IF_MGW_VOCAN_FUNCTION

Method	Function
CREATE_PARAMETER	Creates parameter

1.1.5.2.3.5 /IWBEP/IF_MGW_VOCAN_FUN_PARAM

OData vocabulary annotation interface and methods.

For information about how function parameters are used, see [/IWBEP/IF_MGW_VOCAN_FUNCTION](#).

Methods

Table 1: Methods available for /IWBEP/IF_MGW_VOCAN_FUNCTION

Method	Function
CREATE_FUNCTION	Creates a function
CREATE_LABELED_ELEMENT	Creates a labeled element
CREATE_SIMPLE_VALUE	Create a simple value

1.1.5.2.3.6 /IWBEP/IF_MGW_VOCAN_LABEL_ELEM

OData vocabulary annotation interface and methods

You can create labeled elements from annotations and functional parameters.

Example

DATA:

```
lo_ann_target TYPE REF TO /iwbsp/if_mgw_vocan_ann_target, " Vocabulary Annotation Target

lo_annotation TYPE REF TO /iwbsp/if_mgw_vocan_annotation, " Vocabulary Annotation

lo_label_elem TYPE REF TO /iwbsp/if_mgw_vocan_label_elem. " Vocabulary Annotation Labeled Element

lo_ann_target = vocab_anno_model->create_annotations_target( 'EPMDEMO.PurchaseOrderItem' ).

lo_annotation = lo_ann_target->create_annotation( iv_term = 'org.example.display.DisplayName' ).

lo_label_elem = lo_annotation->create_labeled_element( 'CustomerFirstName' ).
```

In the metadata document, this is reflected as follows:

```
<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="EPMDEMO.PurchaseOrderItem">

    <Annotation Term="org.example.display.DisplayName">

        <LabeledElement Name="CustomerFirstName"/>

    </Annotation>

</Annotations>
```

Methods

Table 1: Methods available for /IWBEP/IF_MGW_VOCAN_LABEL_ELEM

Method	Function
CREATE_ANNOTATION	Creates an annotation
CREATE_COLLECTION	Creates a collection
CREATE_RECORD	Creates a record
CREATE_FUNCTION	Creates a function
CREATE_SIMPLE_VALUE	Creates a simple value
CREATE_URL	Creates a URL

1.1.5.2.3.7 /IWBEP/IF_MGW_VOCAN_PROPERTY

OData vocabulary annotation interface and methods.

A property can be created for a record. For more information, see [/IWBEP/IF_MGW_VOCAN_RECORD](#).

Methods

Table 1: Methods available for /IWBEP/IF_MGW_VOCAN_PROPERTY

Method	Function
CREATE_ANNOTATION	Creates an annotation
CREATE_COLLECTION	Creates a collection
CREATE_RECORD	Creates a record
CREATE_FUNCTION	Creates a function
CREATE_SIMPLE_VALUE	Creates a simple value
CREATE_URL	Creates a URL

1.1.5.2.3.8 /IWBEP/IF_MGW_VOCAN_RECORD

OData vocabulary annotation interface and methods.

A record can be created for an annotation, property, labeled element or collection. The following example shows how a record is created from an annotation. This example also includes the creation of a property for the record.

Example

DATA:

```

lo_ann_target TYPE REF TO /iwbsp/if_mgw_vocan_ann_target, " Vocabulary Annotation Target

lo_annotation TYPE REF TO /iwbsp/if_mgw_vocan_annotation, " Vocabulary Annotation

lo_record      TYPE REF TO /iwbsp/if_mgw_vocan_record,      " Vocabulary Annotation Record

lo_property    TYPE REF TO /iwbsp/if_mgw_vocan_property.    " Vocabulary Annotation Property


lo_ann_target = vocab_anno_model->create_annotations_target( 'Products' ).

lo_annotation = lo_ann_target->create_annotation( iv_term = 'Org.OData.Core.V1.OptimisticConcurrencyControl' ).

lo_record      = lo_annotation->create_record( iv_record_type = 'Some Record Type' ).

lo_property    = lo_record->create_property( iv_property_name = 'My Property' ).

```

In the metadata documentation, this is reflected as follows:

```

<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="Products">

  <Annotation Term="Org.OData.Core.V1.OptimisticConcurrencyControl">

    <Record Type="Some Record Type">

      <PropertyValue Property="My Property"/>

    </Record>

  </Annotation>

</Annotations>

```


Methods

Table 1: Methods available for /IWBEP/IF_MGW_VOCAN_RECORD

Method	Function
CREATE_ANNOTATION	Creates an annotation
CREATE_PROPERTY	Creates a property (PropertyValue)