How to Fine-Tune the Execution of SADL-Based Gateway Services Using the Query Options API
# Content

1. **Using Query Options for Executing SADL-Based Gateway Services.** .............................................. 4
2. **Document History.** .......................................................... 5
3. **Availability of Features in Support Packages.** .......................................................... 6
4. **Prerequisites.** ................................................................. 7
   4.1 Related Documents. .................................................. 7
5. **Background Information.** .................................................. 8
   5.1 Query Options. .......................................................... 8
   5.2 Query Pushdown. ...................................................... 8
6. **Step-by-Step Procedure.** .................................................. 9
   6.1 Access the Query Options. .......................................... 9
   6.2 Restrict the Attributes Returned by the Service. ............... 10
   6.3 Set the Default Sort Criteria for the Query Results. ........... 11
   6.4 Switch Off Language-Specific Sorting to Improve Performance. ........................................ 12
   6.5 Set the Filter Conditions. ........................................... 12
      - Set Range Filter Conditions. ..................................... 14
      - Set Filter Condition Postfix Format. ........................... 15
   6.6 Set the Maximum Number of Returned Results and the Starting Row. ................................ 16
   6.7 Set the Text Search Scope. .......................................... 17
   6.8 Set the Text Search Term. ........................................... 18
   6.9 Set the Threshold Similarity Value for Fuzzy Search. ........... 18
   6.10 Sort Result by Fuzzy Search Score. ................................ 19
   6.11 Restrict the Query Results by Business Criteria. .............. 20
   6.12 Set the Entity Parameters for Core Data Services (CDS) and External Views. ...................... 21
      - Entity Parameters and Navigation. ................................ 22
      - Entity Parameters and Property Mapping with Association Paths. ........................................ 23
   6.13 Restrict the Query Results by User Authorizations. .......... 24
   6.14 Specify the Aggregation Method for Analytical Measures. .................................................. 24
   6.15 Set the Aggregation Count Field. ................................ 25
   6.16 Handling OData Navigation. ......................................... 26
   6.17 Replacing Keys for get_entity Calls. ................................ 28
7. **Support for Customer Extensions.** ........................................ 30
   7.1 Extend OData Model with New Entity Properties. .................. 30
   7.2 Map Entity Properties to Extension Fields. ........................................ 30
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8.1</td>
<td>Background Information</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>8.2</td>
<td>$expand Performance</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>8.3</td>
<td>Decide Whether the Optimized SADL $expand Can Be Used</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>8.4</td>
<td>Redefining the Expand Methods of the Extension DPC</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>8.5</td>
<td>Optimized $expand with SADL and Non-SADL Entities</td>
<td>34</td>
</tr>
<tr>
<td>9</td>
<td>9.1</td>
<td>Terms</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>9.2</td>
<td>Support and Issue Reporting</td>
<td>38</td>
</tr>
</tbody>
</table>
1 Using Query Options for Executing SADL-Based Gateway Services

This document provides a detailed guide on how to use the query options in order to parameterize and fine-tune the execution of your SADL-based SAP Gateway services. These features can be used individually or in combination.
# Document History

Table 1:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4 SP05 V1</td>
<td>First official release of this guide, valid for SAP NetWeaver 7.4 SP05</td>
</tr>
<tr>
<td>7.4 SP07 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP07</td>
</tr>
<tr>
<td>7.4 SP08 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP08</td>
</tr>
<tr>
<td>7.4 SP09 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP09</td>
</tr>
<tr>
<td>7.4 SP10 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP10</td>
</tr>
</tbody>
</table>
3 Availability of Features in Support Packages

The following features are available since the specified support package levels:

SAP NetWeaver 7.4 SP07

- Set the Filter Conditions [page 12]
- Define aggregation and grouping options (chapter Set the Aggregation Count Field [page 25] and Specify the Aggregation Method for Analytical Measures [page 25])

SAP NetWeaver 7.4 SP08

- Performance optimization by switching off language-specific sorting (chapter Switch Off Language-Specific Sorting to Improve Performance [page 12])
- Sort result by fuzzy search score (chapter Set the Threshold Similarity Value for Fuzzy Search [page 18])
- Use CDS view parameters (chapter Set the Entity Parameters for Core Data Services (CDS) and External Views [page 21])
- Support for Customer Extensions [page 30]

SAP NetWeaver 7.4 SP09

- Set Filter Condition Postfix Format [page 15]

SAP NetWeaver 7.4 SP10

- Handling OData Navigation [page 26]
- Replacing Keys for get_entity Calls [page 28]
- Improving OData $expand Performance [page 32]
4 Prerequisites

Before performing this tutorial, take note of the following prerequisites:

- For Gateway Service creation, the reader is kindly referred to the How-To Guide for Creating Gateway Services Based on SADL Business Models.
- The SADL integration is provided for SAP NetWeaver 7.4 SP 5 and higher.

4.1 Related Documents

This document is provided in the SAP Community Network (SCN), which is the social network for SAP professionals.

In addition, you will also find the following SADL-related documents in SCN:

- Enforce Authorizations for SADL-Based Applications
- Model a Gateway Service Based on Business Entities

Note

The related documents might be relevant to each other. This means, for example, if you want to execute the tutorial to model a Gateway service based on business entities, you need to consider the authorization concept.
5 Background Information

5.1 Query Options

The selection criteria used for restricting the amount of data passed to a program or to the user are called query options. They can be defined by:

1. The user who actively sets the selection criteria
2. The application through the setting of criteria as required by the business logic – for example, selecting only approved purchase orders for release
3. The system environment – for example, selecting only data for which the user is authorized

The three categories of selection criteria are used in combination, and only datasets that satisfy all criteria are returned to the caller.

5.2 Query Pushdown

SADL Query enables fast read access for scenarios on mobile and desktop applications based on HANA by means of a query pushdown.

As part of the query pushdown the user’s entire input is collected through consumer APIs and used to configure the request for the database. In addition, the application uses the same consumer APIs to set the business restrictions and the environment parameters for authorization restrictions. The SADL engine combines both categories of inputs and adds the resulting restriction to the condition for the database select (WHERE clause).

As a result, only matching datasets are requested from the database, which results in a reduction both in the load on the database and in the traffic between the database and the application.
6  Step-by-Step Procedure

Context

Note

In the following procedures, the names of the Gateway service properties (for example, `PRODUCT_ID`, `CURRENCY_CODE`, and so on) are the ABAP field names from the SAP Gateway Service Builder. This means that if a parameter contained in the following methods refers to the name of a Gateway service property, you have to take the ABAP field name into consideration.

In the given source code examples, the properties are usually called `elements`.

6.1  Access the Query Options

In the context of SADL-based Gateway services, the consumer API for setting the query options is provided by an object of the type `if_sadl_gw_query_options`.

Context

This object is passed as a parameter of the method `if_sadl_gw_query_control ~ set_query_options` in the generated `_dpc` classes of the SADL-based Gateway services.

```plaintext
METHODS set_query_options
  IMPORTING
  iv_entity_set TYPE string
  io_query_options TYPE REF TO if_sadl_gw_query_options

Applications can use this object in order to read the selection criteria set by the user, or to set their own query options by keeping or overwriting the user query options.
```
In order to do this, perform the following steps:

**Procedure**

1. **Redefine the set_query_options in the generated */_dpc_ext class of your service.**

   ```plaintext
   PUBLIC SECTION.
   METHODS if_sadlgw_querycontrol-set_query_options REDEFINITION.
   ```

2. **Implement the set_query_options method.** You will most likely want to embed the option setting in a case statement in order to set specific query options for each entity set of your service.

   ```plaintext
   METHOD if_sadlgw_querycontrol-set_query_options.
   CASE iv_entity_set.
   WHEN 'Customers'.
   "see the next chapters to find query options that can be set here"
   WHEN 'Products'.
   "see the next chapters to find query options that can be set here"
   WHEN OTHERS.
   END_CASE.
   ENDMETHOD.
   ```

   **Note**

   These steps are the prerequisite for using any of the query options described in the next chapters.

6.2 **Restrict the Attributes Returned by the Service**

You can restrict the properties in each dataset returned by the service.

**Context**

This is especially recommended in cases where the service is meant to be reused in different scenarios, and hence contains more elements than are needed for each usage. Reducing the number of requested elements will directly reduce the load on the database.

To set the requested elements, perform the following steps:

**Procedure**

1. **Define the list of requested elements.**
Note

Use the ABAP field names of the properties.

DATA lt_requested_elements TYPE if_sadl_public_types=>tt_requested_elements.
  lt_requested_elements = VALUE #( ( `PRODUCT_ID` ) ( `PRICE` ) ).

2. Alternatively, you can also get the complete list of elements from the service and delete unnecessary elements from there.

   io_query_options -> get_requested_elements ( IMPORTING et_elements = lt_requested_elements ).
   DELETE lt_requested_elements WHERE table_line = `SUPPLIER_ID`.

3. Set the requested elements.

   io_query_options->set_requested_elements( lt_requested_elements ).

6.3 Set the Default Sort Criteria for the Query Results

You can influence the sort order for the returned datasets by specifying one or more properties according to which the results should be sorted and by specifying the sort direction for each property.

Context

Perform the following steps:

Procedure

1. Get the sorting criteria set by the user. This step is optional; however, if it is not performed, the sort criteria set by the client will always be overwritten.

   DATA lt_sort_elements TYPE if_sadl_public_types=>tt_sort_elements.
   io_query_options->get_sort_elements( IMPORTING et_sort_elements = lt_sort_elements ).

2. Define (additional) sorting criteria and add them to the sort criteria.

   Note

   Use the ABAP field names of the properties.

   APPEND VALUE #( element_name = `PRODUCT_ID` descending = abap_true ) to lt_sort_elements.
3. Set the sort criteria.

\[
\text{io_query_options}\rightarrow\text{set_sort_elements( lt_sort_elements )}.
\]

6.4 Switch Off Language-Specific Sorting to Improve Performance

SADL runtime uses language-specific alphabetical sorting as the default sort strategy. If sort criteria are defined, the records in the result are sorted according to the alphabet of current user’s language.

Prerequisites

This feature is available since SAP NetWeaver 7.40 SP08.

Context

The disadvantage of using language-specific sorting is that the database might need much more time to execute the query, even with factor 10 or more. To protect performance, you can switch the language-specific sorting off, if it is not needed. It is recommended that you use language-specific sorting only if:

- $orderby is defined on a text field or
- $filter with operators ‘gt’, ‘ge’, ‘lt’ or ‘le’ is defined on a text field.

The text field is a database field that contains language-specific texts. In all other cases you can switch the language-specific sorting off by calling the following method:

\[
\text{io_query_options}\rightarrow\text{set_locale_disabled( abap_true )}.
\]

6.5 Set the Filter Conditions

Filter conditions set by the OData client can be read and modified by, the application.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP09. Prior to SP09, only the range tables were accessible to the applications, as described in the following chapter.
Context

Perform the following steps:

Procedure

1. **Retrieve the filter conditions in postfix format.**

   ```
   Note
   The filter refers to the ABAP field names of the properties.
   ```

   ```
   io_query_options->get_filter_condition( IMPORTING et_filter_condition = DATA(lt_filter_condition) ).
   ```

2. **Modify the filter, for example by adding a condition. The code adds the condition "and SUPPLIER_ID eq '100000002'" to the client filter:**

   ```
   APPEND VALUE #( element = 'SUPPLIER_ID' operator = 'EQ' low = `100000002` ) TO lt_filter_condition.
   IF lines( lt_filter_condition ) > 1.
   APPEND VALUE #( operator = 'AND' ) TO lt_filter_condition.
   ENDIF.
   ```

3. **Set the filter conditions. This overwrites the filter set by the consumer.**

   ```
   io_query_options->set_filter( lt_filter ).
   ```

   ```
   Note
   The parameter lt_filter_condition has to be used in this case. For compatibility reasons the method set_filter expects – by default – a filter condition in the format of range table, which is less flexible. A filter condition either in the range tables format or in the postfix format can be used to overwrite the client filter. If you try to set two conditions in different formats at once, an exception of type cx_sadi_gw_contractViolation is raised.
   ```

Related Information

Set Filter Condition Postfix Format [page 15]
6.5.1 Set Range Filter Conditions

Simple filter conditions set by OData client can mostly be expressed by range tables. If this is the case, the application can use this format to read and modify the client filter.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP07.

Context

This format can be used alternatively to the postfix conditions described above.

Perform the following steps:

Procedure

1. Retrieve the filter in the format of range tables.

   ```
   io_query_options->get_filter( IMPORTING et_filter = DATA(lt_filter) ).
   ```

   **Note**

   The filter refers to the ABAP field names of the properties.

2. Read and modify the filter ranges.

   ```
   " exclude products with supplier 100000002
   READ TABLE lt_filter WITH KEY name = 'SUPPLIER_ID'
   ASSIGNING FIELD-SYMBOL(<s_filter_range>).
   IF sy-subrc = 0.
     APPEND VALUE #( sign = 'E' option = 'EQ' low = '100000002' )
     TO <s_filter_range>-range.
   ELSE.
     INSERT VALUE #( name = 'SUPPLIER_ID'
   ```

   **Note**

   The method `get_filter` raises a static exception of type `cx_sadl_gw_filter_condition` if the filter set by the OData client cannot be expressed by ranges - for example, a filter string "SupplierId eq '100000000' or SupplierName eq 'SAP'". In such a case, only the postfix format can be used, as described in the previous chapter. The `get_filter` method raises a static exception of `cx_sadl_gw_filter_condition` type if the filter set by the OData client cannot be expressed by ranges – for example, a filter string "SupplierId eq '100000000' or SupplierName eq 'SAP'". In such a case, only the postfix format can be used, as described in the previous chapter.
3. Set the filter ranges to overwrite the filter set by the OData client.

```plaintext
io_query_options->set_filter( lt_filter ).
```

**Note**

If the application needs to extend the condition in a way that cannot be expressed as ranges, it is recommended that you use the methods for the postfix format.

### Related Information

Set Filter Condition Postfix Format [page 15]

### 6.5.2 Set Filter Condition Postfix Format

Postfix format for filter conditions allows us to combine conditions arbitrarily with logical operators \AND, OR, and \NOT. Postfix means that operands are followed by an operator.

In our case, two conditions are combined with the logical operator \AND or \OR if the conditions are followed by that operator.

A condition is negated if it is followed by the operator \NOT. The resulting conditions can be combined by the operators in the same manner.

**Examples**

Condition as a string in infix format expressed as a filter condition:

- **Condition**: price < 100 and currency_code = 'EUR'

  ```plaintext
  it_filter_condition = value #(
    ( element = 'PRICE' operator = 'LT' low = '100' )
    ( element = 'CURRENCY_CODE' operator = 'EQ' low = 'EUR' )
    ( operator = 'AND' )
  )
  ```

- **Condition**: category = 'Notebooks' or category = 'Tablets'

  ```plaintext
  it_filter_condition = value #(
    ( element = 'CATEGORY' operator = 'EQ' low = 'Notebooks' )
    ( element = 'CATEGORY' operator = 'EQ' low = 'Tablets' )
    ( operator = 'OR' )
  )
  ```

- **Condition**: price < 100 and currency_code = 'EUR' and (category = 'Notebooks' or category = 'Tablets')

  ```plaintext
  it_filter_condition = value #(
    ( element = 'PRICE' operator = 'LT' low = '100' )
    ( element = 'CURRENCY_CODE' operator = 'EQ' low = 'EUR' )
    ( operator = 'AND' )
  )
  ```
6.6 Set the Maximum Number of Returned Results and the Starting Row

The service consumers can specify the maximum number of expected results for a call as well as the number of datasets to be skipped.

Context

If nothing is specified by the consumer, all records to be skipped will be selected and no records will be skipped.

You can override the user settings, or set your own defaults by performing these steps:

Procedure

1. Retrieve the values set by the consumer or framework.

   ```
   io_query_options->get_paging( IMPORTING ev_skip = DATA(lv_skip)
                                ev_top = DATA(lv_top) ).
   ```

2. Set your own values by passing the paging query option to the service.

   ```
   io_query_options->set_paging( iv_skip = lv_skip iv_top = lv_top ).
   ```

   **Note**

   The `iv_skip` variable sets the number of records to be skipped; the starting row is therefore `iv_skip + 1`.

   Both `iv_skip` and `iv_top` apply to the sorted, aggregated, and restricted lists after all other query options have been applied.
6.7 Set the Text Search Scope

If your service is to support a text search function, you need to set the search scope – that is, the properties across which the search is to be performed.

Context

For example, you might want to search for a name in the Product Name (NAME) and Category Name (CATEGORY_NAME) columns, but not in the Supplier Name (SUPPLIER_NAME).

To set the list of properties where the search is to be executed, perform the following steps:

Procedure

1. Define the search scope:

   Use the ABAP field names of the properties.

   ```abap
   DATA lt_search_scope TYPE if_sadl_public_types=>tt_search_scope.
   lt_search_scope = VALUE #( 'CATEGORY_NAME' ( 'NAME' ) ).
   ```

2. Pass the search scope to the service.

   ```abap
   io_query_options->set_text_search_scope( lt_search_scope ).
   ```

   Leaving the search scope empty will result in runtime exceptions if the consumer attempts a search. Not all data types support the search for each database. With SAP HANA, the text search is not supported on columns of type string or raw.
6.8  Set the Text Search Term

You can retrieve the search term set by the consumer and enhance or replace it with your own defaults.

Context

You can also use wildcards for the search. The engine will try to match the entire specified search term against values in one of the columns contained in the search scope.

In order to set the search term:

Procedure

1. Set the search scope, as described in the previous chapter.
2. Retrieve the search term.
   
   \[
   \text{DATA(lv\_search\_term) = io\_query\_options \rightarrow get\_text\_search\_term(\)}.\]
3. Set the new or default search term.
   
   \[
   \text{lv\_search\_term = 'myterm*'&& lv\_search\_term.} \\
   \text{io\_query\_options\rightarrowset\_text\_search\_term(lv\_search\_term).}\]

6.9  Set the Threshold Similarity Value for Fuzzy Search

You can indicate that you want a fuzzy (inexact, approximate) search to be performed.

Prerequisites

This kind of search will only be performed on databases that support fuzzy search capabilities.

Context

You can further specify the search accuracy by setting the value for a fuzzy threshold similarity to a value between 0.9 (high accuracy) and 0.1 (low accuracy, many matches). Both 1.0 and 0.0 will result in exact searches to be performed.
Procedure

Define the similarity value and pass it to the service.

```
DATA lv_similarity TYPE if_sadl_public_types=>ty_similarity VALUE '0.7'.
io_query_options->set_text_search_similarity( lv_similarity ).
```

6.10 Sort Result by Fuzzy Search Score

If you use fuzzy search, you might want to sort records in the result according to relevance. The relevance is expressed as a number called score.

Prerequisites

This feature is available since SAP NetWeaver 7.40 SP08.

Context

Usually you want the records that fully match the search term to be on top (score = 1), the records with partial match to be further down (score < 1). To achieve this, you have to:

Procedure

1. Define a virtual name for the score (called score alias), which you can use as the element name in the definition of the sort criteria. Note that you may define the alias only if a search term is provided.
2. Extend the sort criteria defined by the client by appending a criterion to the score alias. Optionally, for better performance you can provide the sort criterion for the score only if the client does not define any other sort criteria.

Results

The code would look like this:

```
IF  io_query_options->get_text_search_term ( ) IS NOT INITIAL.
  io_query_options->set_text_search_score_alias( 'SEARCH_SCORE' ).
  io_query_options->get_sort_elements( 
    IMPORTING et_sort_elements = DATA(lt_sort_elements) ).
```
APPEND VALUE #( element_name = 'SEARCH_SCORE' descending = abap_true ) TO lt_sort_elements.
io_query_options->set_sort_elements( lt_sort_elements ).
ENDIF.

Note
The score alias does not have to be a property of the OData entity. But if needed, you can also expose the score to the OData client. To do this, add a property to the entity type and use the score alias as the ABAP field name of this property. The type of the property will be Edm.Decimal with precision 3, scale 1. Leave the property mapping to the data source unassigned.

6.11 Restrict the Query Results by Business Criteria

The consumer can specify selection criteria for the query results. In addition, applications can define their own default selection criteria as required by the business process. You can do this by creating, parameterizing, and providing a condition provider.

Context

Query condition providers are objects that implement the interface if_sadl_condition_provider. For practical purposes, SADL provides a condition provider factory that creates condition provider instances for given conditions, and groups the combination of different basic condition providers into one complex condition provider using Boolean functions (and, or, not).

In the following example, the service is to be restricted by a complex condition to return only products with a dimension unit meter (m) or a weight unit kilogram (kg).

Note
You can repeat the following steps with another condition provider. Conditions from all providers added to the query options using the method add_condition_provider will be applied to the result, equivalent to an AND between condition providers.

You can also directly use one of the basic condition providers in the method. This will result in the application of only the respective basic condition to the returned datasets.

Procedure

1. Get an instance of the basic condition provider factory.

DATA(lo_condition_factory) =
cl_sadl_cond_prv_factory_pub=>create_basic_condition_factory( ).
2. Create basic condition providers for all the conditions you want to set.

   **Note**
   
   Use the ABAP field names of the properties for defining the field mapping.

   ```
   DATA(lo_cond_dim_unit) = lo_condition_factory->equals(
     name = 'DIM_UNIT' "use the ABAP Field Name of a property
     value = 'M' ).
   ```

   ```
   DATA(lo_cond_weight_unit) = lo_condition_factory->equals(
     name = 'WEIGHT_UNIT'
     value = 'KG' ).
   ```

3. Combine the basic condition providers using Boolean functions.

   ```
   DATA(lo_condition_provider) = lo_condition_weight_unit->or( lo_condition_dim_unit).
   ```

4. Add the resulting condition provider to the service.

   ```
   io_query_options->add_condition_provider( lo_condition_provider ).
   ```

6.12 Set the Entity Parameters for Core Data Services (CDS) and External Views

**Prerequisites**

The CDS view parameters can be used since SAP NetWeaver 7.40 SP08.

**Context**

Gateway services constructed on the basis of:
- SAP HANA views with parameters or
- CDS views with parameters

need to provide values for these parameters before selection. Parameters for all related views with parameters have to be provided in the same method call. See the following sections.

In a simple case, you can pass these parameters using the query options with the following steps:

**Procedure**

1. Set the values for the parameters.
i Note

Use the entity set name for `entity_alias`. If you are using CDS, you have to enter the parameters in the same order as declared in the CDS definition.

```abap
DATA lt_parameters TYPE if_sadl_public_types=>tt_entity_parameters.
  APPEND VALUE #( entity_alias = 'SalesOrderHeaders'
                parameters = VALUE #( ( name = 'CURRENCY_CODE' value = 'EUR' )
                                     ( name = 'STATUS' value = 'APPROVED' ) )
                 )
    TO lt_parameters.

2. Pass the parameter table to the service.

io_query_options->set_entity_parameters( lt_parameters ).
```

i Note

CDS views with parameters are not supported by all databases. To check whether a database supports the CDS views with parameters, refer to the SAP Product Availability Matrix (PAM), or call in your application:

```abap
rv_supported = cl_abap_dbfeatures=>use_features( 
    requested_features = VALUE #( ( cl_abap_dbfeatures=>views_with_parameters ) )
).
```

### 6.12.1 Entity Parameters and Navigation

If an OData entity is accessed through navigation, parameters for the principle entity have to be provided as well.

#### Example 1

Let us suppose the OData client navigates from `SalesOrderHeaders` to `SalesOrderItems` with a navigation path like:

```plaintext
.../SalesOrderHeaders('500000000')/Items
```

If entity parameters are supplied conditionally, depending on the requested entity data, the parameters for `SalesOrderHeaders` will have to be provided as well whenever data from `SalesOrderItems` is requested:

```abap
IF iv_entity_set = 'SalesOrderHeaders'
  OR iv_entity_set = 'SalesOrderItems', " principal entity parameters needed
  io_query_options->set_entity_parameters( lt_parameters = VALUE #( ( entity_alias = 'SalesOrderHeaders'
                                                                   parameters = VALUE #( ( name = 'CURRENCY_CODE' value = 'EUR' )
                                                                   ( name = 'STATUS' value = 'APPROVED' ) )
                                                                   )
                     )
```
Example 2

Let us suppose the OData client navigates from `SalesOrderHeaders` to `SalesOrderItems`, like in the previous example, and that `SalesOrderItems` is based on a view with parameters. Parameters for both entities will have to be provided at once:

```csharp
io_query_options->set_entity_parameters( it_parameters = VALUE #(
    ( entity_alias = 'SalesOrderItems'
        parameters = VALUE #( ( name = 'LANGUAGE'      value = sy-langu ) )
    ( entity_alias = 'SalesOrderHeaders'
        parameters = VALUE #( ( name = 'CURRENCY_CODE' value = 'EUR' ),
                                  ( name = 'STATUS'      value = 'APPROVED' )
    )
) ).
```

6.12.2 Entity Parameters and Property Mapping with Association Paths

If an OData entity property is mapped to an element of an associated business entity, parameters for this entity have to be supplied also. Use the pattern `<entity set name>.<associated business entity path>` as `entity_alias`.

Example

Here, a property `BusinessPartnerCompanyCountry` of `SalesOrderHeaders` is mapped to an element `BUSINESS_PARTNER.COMPANY.COUNTRY` and that the referenced business entity is based on a view with parameter `LANGUAGE`. The parameter for the view must be provided together with other parameters (if any).

```csharp
io_query_options->set_entity_parameters( it_parameters = VALUE #(
    ( entity_alias = 'SalesOrderHeaders.BUSINESS_PARTNER.COMPANY'
        parameters = VALUE #( ( name = 'LANGUAGE'      value = sy-langu ) )
    ( entity_alias = 'SalesOrderHeaders'
        parameters = VALUE #( ( name = 'CURRENCY_CODE' value = 'EUR' ),
                                  ( name = 'STATUS'      value = 'APPROVED' )
    )
) ).
```

How to Fine-Tune the Execution of SADL-Based Gateway Services

© 2015 SAP SE or an SAP affiliate company. All rights reserved.
6.13 Restrict the Query Results by User Authorizations

Services constructed on the basis of a SADL business model in the Gateway Service Builder can be protected using an authorization provider. Therefore, the application has to decide on a suitable provider and pass an instance of the provider to the Gateway engine.

Context

Perform the following steps:

Procedure

1. Create a suitable authorization provider.
   More on this: How to Enforce Authorizations for SADL-Based Applications.
2. Create an instance of your authorization provider. If you use a generic provider, you also need to pass the authorization parameters for the check.
   
   **Note**
   
   Use the ABAP field names of the properties for defining the field mapping.

   ```
   DATA(lo_provider) = 
   cl_sadl_cond_prov_factory_pub=>create_for_authorization( ).
   lo_provider->add_authorization_for_object(
      iv_authorization_object  = 'S_EPM_PD'
      it_activities           = VALUE #( ( auth_field = 'ACTVT' value = '03' ) )
      it_field_mapping        = VALUE #( ( auth_field = 'PDCATEGORY' view_field = 'CATEGORY' ) )
   ).
   ```
3. Finally, pass the provider object to the Gateway engine.

   ```
   io_query_options->set_authority_provider( lo_provider ).
   ```

   **Note**
   
   Transactional services, unlike the query, delegate the authorization enforcement to the framework that corresponds to their business entities.

   Find out more about the authorization enforcement according to the relevant framework:
   - Authorsizations in BOPF (Business Object Processing Framework)
   - And so on
6.14 Specify the Aggregation Method for Analytical Measures

If your service performs an aggregation, you need to specify aggregation behavior of the defined measures using the query options API. This will have no effect on the OData metadata, that is, it will result in no annotations.

Prerequisites

- This feature is available since SAP NetWeaver 7.4 SP07.
- You have maintained the OData annotations for your dimensions and measures.
- You have mapped the measure property to the field to be aggregated.

Procedure

Set the aggregation fields. Provide the measure to be aggregated as `element` and the ABAP Field Name of the property that contains the result of the aggregation as `alias`.

```plaintext
io_query_options -> set_aggregation ( VALUE #( 
  ( element = 'GROSS_AMOUNT' alias = 'GROSS_AMOUNT' 
    type = if_sadl_gw_query_options=>co_aggregation_type-avg ) ) ).
```

6.15 Set the Aggregation Count Field

If your service performs an aggregation, you can specify an output field that contains the values count corresponding to a specific combination of the aggregation input values.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP07.
Then, define and set the ABAP field name of the property for returning the aggregation count:

```abap
io_query_options->set_count_field_name( 'SALES_ORDERS_COUNT' ).
```

### 6.16 Handling OData Navigation

If an OData association is mapped by SADL to a data source, SADL automatically selects the data following the navigation path from the OData request. Nothing has to be programmed by the application. But if an OData association is not mapped by SADL to a data source, you have to interpret the navigation path yourself.

#### Prerequisites

This feature is available since SAP NetWeaver 7.4 SP10.

#### Context

Typically, if your service contains an application-implemented entity set with an association to an entity set implemented by SADL, you need to provide a condition to select the right target entities.

For example, suppose that the entity set `SalesOrderHeaders` is implemented by the application and the entity set `SalesOrderItems` is implemented by SADL. When a user navigates from `SalesOrderHeaders` to `SalesOrderItems`, you have to call the `remove_navigation_info` method to force SADL to ignore the navigation information. In addition, you have to provide a condition to identify the target entities on the database.

Another example is when you have to replace the navigation information with a condition, where the source or the target entity key is to be replaced by another key of the underlying database view.

Follow these steps to implement your own interpretation of the navigation:

#### Procedure

1. Create class instance attributes to store the navigation information.

   ```abap
   PRIVATE SECTION.
   DATA mt_navigation_info TYPE /iwbep/t_mgw_tech_navi.
   DATA mt_source_keys TYPE /iwbep/t_mgw_tech_pairs.
   ```
2. In the \texttt{getter} methods of the target entity, store the navigation information:

\begin{verbatim}
METHOD salesorderitems_get_entity.
    mt_navigation_info = io_tech_request_context->get_navigation_path( ).
    mt_source_keys     = io_tech_request_context->get_source_keys( ).
    ...
ENDMETHOD.
METHOD salesorderitems_get_entityset.
    mt_navigation_info = io_tech_request_context->get_navigation_path( ).
    mt_source_keys     = io_tech_request_context->get_source_keys( ).
    ...
ENDMETHOD.
\end{verbatim}

3. Force SADL to ignore the navigation and define a condition to select the target entities from the database:

\begin{verbatim}
METHOD if_sadl_gw_query_control_set_query_options.
    CASE iv_entity_set.
        WHEN 'SalesOrderItems'.
            IF mt_navigation_info IS NOT INITIAL.
                DATA(ls_nav_step) = mt_navigation_info[ 1 ].
                " Is this the navigation we have to handle (SO -> Items)?
                IF ls_nav_step-source_entity_type = 'SalesOrderHeader'
                    AND ls_nav_step-nav_prop      = 'ITEMS'.
                    " Make SADL ignore navigation
                    io_query_options->remove_navigation_info().
                " Define condition based on source entity key
                DATA(lo_cond_factory) = cl_sadl_cond_prov_factory_pub=>create_basic_condition_factory( ).
                io_query_options->add_condition_provider(  
                    lo_cond_factory->equals(  
                        name = 'SO_ID' " ABAP field name of GW property  
                        value = mt_source_keys[ name = 'SO_ID' ]-value ) ).
                " Define condition based on target entity key (if any)
                IF ls_nav_step-key_tab IS NOT INITIAL.
                io_query_options->add_condition_provider(  
                    lo_cond_factory->equals(  
                        name = 'SO_ID' " ABAP field name of GW property  
                        value = ls_nav_step-key_tab[ name = 'SO_ID' ]-value ) )  
                    && (  
                        lo_cond_factory->equals(  
                            name = 'SO_ITEM_POS' " ABAP field name of GW property  
                            value = ls_nav_step-key_tab[ name = 'SO_ITEM_POS' ]-value )  
                    )  
                )
                ENDIF.
            ENDIF.
        ENDIF.
    END CASE.
ENDMETHOD.
\end{verbatim}

\textbf{Note}

The following happens if you call the \texttt{remove_navigation_info} method:

\begin{itemize}
    \item In a non-navigation scenario, an exception is raised.
    \item In the case of a navigation that is not mapped by SADL, an exception is raised.
\end{itemize}

\textbf{Related Information}

Replacing Keys for get\_entity Calls [page 28]
6.17 Replacing Keys for get_entity Calls

You can use the `remove_keys()` method to force SADL to ignore an entity key specified by an OData request. You must provide a condition to identify the requested entity on the database.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP10.

Context

A possible scenario for this would be as follows. Suppose that your OData entity type contains one key property, for example - the ID, which is created as a concatenation of multiple fields of the underlying database view. The selection of an entity by an ID would be "expensive" on the database since it would require evaluation of the ID for each and every record in the database. Therefore, you can force SADL to ignore the ID and you have to define a condition for the other fields of the underlying view.

An example implementation would look like this.

Procedure

1. Create a class instance attribute for storing the entity key.

   ```plaintext
   PRIVATE SECTION.
   DATA mt_keys TYPE /iwbep/t_mgw_tech_pairs.
   ```

2. In the `<entitySet>_get_entity` method, store the key from the technical request.

   ```plaintext
   METHOD salesorderitems_get_entity.
   IF io_tech_request_context->get_navigation_path( ) IS INITIAL.
      mt_keys = io_tech_request_context->get_keys( ).
   ENDIF.
   ```

3. Replace the key in the `if_sadl_gw_query_control~set_query_options` method:

   ```plaintext
   CASE iv_entity_set.
   WHEN ' SalesOrderItems'.
      IF mt_keys IS NOT INITIAL.
         " Make SADL ignore key from OData request
         io_query_options->remove_keys( ).
         " Build condition (split key, which was made by concatenation with
         separator `~`)
         DATA(lo_cond_factory) =
         cl_sadl_cond_prov_factory_pub=>create_basic_condition_factory( ).
         SPLIT mt_keys[ I ]-value AT `~` INTO DATA(lv_sales_order_id)
         DATA(lv_position).
   ```
DATA(lo_condition) =
    lo_cond_factory->equals( name = 'SO_ID' " ABAP field name of GW property
        value = lv_sales_order_id ) ->and( 
    lo_cond_factory->equals( name = 'SO_ITEM_POS' " ABAP field name of GW property
        value = lv_position ) )
io_query_options->add_condition_provider( lo_condition ).
ENDIF.
ENDCASE.

**Note**

If the entity is used as a source or target entity within an OData navigation, you must also implement the navigation condition. If you call the remove_keys method within a navigation request, an exception is raised. The remove_navigation_info method has to be called instead.

**Note**

If you call the remove_keys method but do not add a condition restricting the number of target entities to one, an exception is raised.

**Related Information**

Handling OData Navigation [page 26]
7  Support for Customer Extensions

Customers have the option available for extending DDIC transparent tables and CDS views with additional fields – extension fields. To expose data from the extension fields in an OData service, you need to extend the OData model by appending new entity properties to a corresponding entity type. Subsequently, you will need to map the properties to the extension fields.

Note

This feature is available since SAP NetWeaver 7.4 SP08.

7.1  Extend OData Model with New Entity Properties

There are a few options available for appending new properties to an OData entity in a customer system.

One option is to create a new project in the Gateway Service Builder and to redefine the data model of the OData service that you want to extend. You can append new properties to a corresponding entity type in the data model redefinition.

Another option is to extend the MPC_EXT class by redefining the method DEFINE and adding new properties to the entity type. The method can look like this example:

```
METHOD define.
  super->define( ).
  DATA(lo_entity_type) = model->get_entity_type( 'Product' ).
  DATA(lo_property) = lo_entity_type->create_property(    
    iv_property_name = 'HazardWarning',
    iv_abap_fieldname = 'HAZARD_WARNING').
  lo_property->set_type_edm_decimal( ).
  lo_property->set_precision( iv_precision = 3 ).
  lo_property->set_maxlength( iv_max_length = 18 ).
  lo_property->set_creatable( abap_false ).
  lo_property->set_updatable( abap_false ).
  lo_property->set_sortable( abap_false ).
  lo_property->set_nullable( abap_false ).
  lo_property->set_filterable( abap_false ).
ENDMETHOD.
```

7.2  Map Entity Properties to Extension Fields

To map new entity properties to the extension fields, you have to redefine the method IF_SADL_GW_EXTENSION_CONTROL~SET_EXTENSION_MAPPING of the DPC_EXT class. The following
example shows how to map the property HazardWarning (with ABAP name HAZARD_WARNING) to the extension field EXT_TURN_OVER and Profit (with ABAP name PROFIT) to EXT_PROFIT.

METHOD if_sadl_gw_extension_control~set_extension_mapping.
  IF iv_entity_set_name = 'Products'.
    io_extension_mapping->set_property_mapping(
      _it_property_mapping = VALUE #(  
        ( property_abap_name = 'HAZARD_WARNING' " OData Entity  
          business_entity_element = 'EXT_HAZARD_WARNING' ) )  " Data Source  
        ( property_abap_name = 'PROFIT' " OData Entity  
          business_entity_element = 'EXT_PROFIT' ) ) ). " Data Source
  ENDF.
ENDMETHOD.
8 Improving OData $expand Performance

Retrieving data for a request using $expand parameters can be very slow. The following sections explain to you how you can optimize performance.

Related Information

Background Information [page 32]
$expand Performance [page 32]
Decide Whether the Optimized SADL $expand Can Be Used [page 33]
Redefining the Expand Methods of the Extension DPC [page 34]
Optimized $expand with SADL and Non-SADL Entities [page 35]

8.1 Background Information

Using the URI query option $expand it is possible to follow navigation properties. The entries associated with the leading entry/entries by the navigation property must be loaded by the service and are represented inline in the result.

Request URI example in the Gateway client:

/sap/opu/OData/SAP/S_EPM_SADL_GW_DEV_SCEN_RO_SRV/SalesOrderHeaders?$expand=Items

- Identifies the collection of SalesOrderHeaders as well as each of the SalesOrderItems associated with each SalesOrderHeader.
- The SalesOrderItems are associated through the navigation property items of the SalesOrderHeader entity type.

The application does not have to care about the implementation of $expand. This implementation is performed by the Gateway runtime out of the box in a generic way. However, the generic approach has a performance drawback. SADL provides a solution to significantly improve the performance for OData associations mapped by SADL to a data source.

8.2 $expand Performance

In the standard $expand implementation, several single roundtrips are executed by the Gateway runtime.

For each single SalesOrderHeader entity, the association to the corresponding items is followed. This means that for n leading entities, n database selections are executed for each navigation step specified in the
$expand. In the optimized SADL $expand only one database selection is executed for each specified navigation step. However, the expand is optimized only for OData associations mapped by SADL to a data source. If an association is not mapped by SADL, the standard Gateway runtime is used.

8.3 Decide Whether the Optimized SADL $expand Can Be Used

The optimized SADL $expand is not provided out of the box by the generated DPC. The SADL $expand must be triggered by the application in the DPC_EXT class.

If you use the optimized SADL $expand, you have to be aware that the following methods of the DPC_EXT class are not called whenever an OData request contains the $expand clause:

- `<entitySet>_get_entity`
- `<entitySet>_get_entityset`
- `/iwbep/if_mgw_appl_srv_runtime~get_entity`
- `/iwbep/if_mgw_appl_srv_runtime~get_entity_set`

If you do not redefine any of the methods listed above in your the DCP_EXT class, you can use the optimized SADL $expand as described in the Redefining the Expand Methods of the Extension DPC [page 34] chapter.

If you do redefine any of the methods listed above, you can use the optimized SADL $expand only provided you are able to implement the same logic in the `/iwbep/if_mgw_appl_srv_runtime~get_expanded_entity` and `/iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset` methods. The following DPC_EXT methods are called if the optimized SADL $expand is used:

- One of the `/iwbep/if_mgw_appl_srv_runtime~get_expanded_entity` or `/iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset` methods is called once by Gateway for the leading entity set.
- The `if_sadl_gw_query_control~set_query_options` method is called once by the SADL runtime for the leading entity set and once for each navigation step specified in the $expand.

If you cannot implement the logic from the getter methods in the `get_expanded_entity` and `get_expanded_entityset`, the optimized SADL $expand cannot be used.
8.4 Redefining the Expand Methods of the Extension DPC

If all entity sets of a Gateway service are mapped to business entities and the restriction of the optimized SADL $expand can be accepted by the application, as discussed in the previous chapter, the optimized SADL $expand can be used.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP10.

Context

If some of the entity sets in your OData service are not mapped to a business entity, see how you can support navigation in a hybrid scenario in combination with the optimized expand in the next section.

To optimize $expand, follow these steps:

Procedure

1. Redefine the get_expanded_entity and get_expanded_entityset methods in the generated *_dpc_ext class of your service:

   ```
   PUBLIC SECTION.
   METHODS /iwbep/if_mgw_appl_srv_runtime~get_expanded_entity REDEFINITION.
   METHODS /iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset REDEFINITION.
   ```

2. Implement the get_expanded_entity method.

   ```
   METHOD /iwbep/if_mgw_appl_srv_runtime~get_expanded_entity.
   " -------------------------------------------------
   " ... Implement application specific logic of get_entity methods here (if any)
   " ... -------------------------------------------------
   " Read entity data (and expand):
   if_sadl_gw_dpc_util~get_dpc( )->get_expanded_entity(  
      EXPORTING io_expand_node = io_expand  
      io_tech_request_context = io_tech_request_context
      IMPORTING er_entity = er_entity  
      et_expanded_tech_clauses = et_expanded_tech_clauses  
      es_response_context = es_response_context ).
   ENDMETHOD.
   ```

3. Implement the get_expanded_entityset method.

   ```
   METHOD /iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset.
   ```
8.5 Optimized $expand with SADL and Non-SADL Entities

If your service combines entity sets that are not mapped by SADL to a data source (non-SADL entities) and entity sets mapped by SADL (SADL entities), you can optimize $expand for associations mapped by SADL, if any exist.

**Context**

However:

- You have to handle navigation manually.
- You have to store the navigation information in the `get_expanded_entity` and `get_expanded_entityset` methods because the standard getter methods are not called.
- You have to call the superclass `get_expanded_entity` and `get_expanded_entityset` methods to enable the default implementation of `$expand` by the Gateway runtime.

**Procedure**

1. Follow the steps described in the **Handling OData Navigation** [page 26] section.
   1. Create class instance attributes to store the navigation information.
   2. Save the navigation information in the `get_entity` and `get_entityset` methods.
   3. Interpret the navigation in the `set_query_options` method.
   2. Remember the navigation information and call the superclass method in the method `get_expanded_entity`.

   ```
   METHOD /iwbsp/if_mgw_appl_srv_runtime-get_expanded_entity.
   " -------------------------------
   " mt_navigation_info = io_tech_request_context->get_navigation_path( ).
   " mt_source_keys = io_tech_request_context->get_source_keys( ).
   " ...
   " -------------------------------
   " Read entity data (and expand), if mapped by SADL
   if_sadl_gw_dpc_util-get_dpc( )->get_expanded_entityset( 
     EXPORTING io_expand_node = io_expand
     io_tech_request_context = io_tech_request_context
     IMPORTING er_entityset = er_entityset
     et_expanded_tech_clauses = et_expanded_tech_clauses
     es_response_context = es_response_context ) .
   ```
3. Remember the navigation information and call the superclass method in the `get_expanded_entityset` method:

```java
METHOD /iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset.
  " -------------------------------------------------
  mt_navigation_info = io_tech_request_context->get_navigation_path( ).
  mt_source_keys     = io_tech_request_context->get_source_keys( ).
  " ...
  " -------------------------------------------------
  " Read entity data (and expand), if mapped by SADL
  if_sadl_gw_dpc_util~get_dpc( )->get_expanded_entityset(
    EXPORTING io_expand_node = io_expand
    io_tech_request_context = io_tech_request_context
    IMPORTING er_entityset = er_entityset
    et_expanded_tech_clauses = et_expanded_tech_clauses
    es_response_context = es_response_context
    ev_entity_mapped_by_sadl = DATA(lv_entity_mapped) ).
  " Entity sets not mapped to a Business Entity are handled as usual by the DPC
  IF lv_entity_mapped <> abap_true.
    super->/iwbep/if_mgw_appl_srv_runtime~get_expanded_entityset(
      EXPORTING iv_entity_name = iv_entity_name
      iv_entity_set_name = iv_entity_set_name
      iv_source_name = iv_source_name
      it_navigation_path = it_navigation_path
      it_key_tab = it_key_tab
      io_expand = io_expand
      io_tech_request_context = io_tech_request_context
      IMPORTING er_entityset = er_entityset
      et_expanded_clauses = et_expanded_clauses
      et_expanded_tech_clauses = et_expanded_tech_clauses
      es_response_context = es_response_context
    ).
  ENDIF.
ENDMETHOD.
```
Related Information

Decide Whether the Optimized SADL $expand Can Be Used [page 33]
Handling OData Navigation [page 26]
9 Appendix

9.1 Terms

Table 2:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Entity</td>
<td>• Entities that are used in business applications – for example, business objects, database tables, search views.</td>
</tr>
<tr>
<td></td>
<td>• Business model metadata interface that offers a harmonized metadata consumption of existing and new business models. The interface is implemented for BOPF and DDIC tables and views productively.</td>
</tr>
<tr>
<td>Business Entity type (or source)</td>
<td>Specific repository or provider for Business Entity metadata – for example, Business Object Processing Framework, Data Dictionary, Core Data Services.</td>
</tr>
</tbody>
</table>

9.2 Support and Issue Reporting

If a functional error with the SADL runtime occurs, report an incident for the BC-ESI-ESF-BSA application component.

If you have other technical issues, check the SAP Communication Network (SCN) →.
Important Disclaimers and Legal Information

Coding Samples

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

Accessibility

The information contained in the SAP documentation represents SAP’s current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of wilful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with “you”, or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or wilful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).