How to Enforce Authorizations for SADL-Based Applications
Using Authorization Providers
# Content

1. **Authorizations for SADL-Based Applications.** ........................................... 3
2. **Document History.** .................................................................................. 4
3. **Prerequisites.** ............................................................................................ 5
4. **Background Information.** ........................................................................ 6
   4.1 Authorization Enforcement. .................................................................. 6
      Authority Check in ABAP. .................................................................. 6
      Other Authorization Mechanisms. .................................................. 6
      Authority Check in Query Push-down. ......................................... 7
      Authority Check in Transactional Scenarios. ............................... 7
   4.2 Authorization Condition Providers. ...................................................... 7
      Modeled Authorizations. ................................................................ 7
      Application-Defined Authorizations. .......................................... 8
5. **Step-by-Step Procedure.** ........................................................................ 9
   5.1 Analysis of Authorization Requirements. .......................................... 9
   5.2 Default Authorization Providers. ....................................................... 9
      Default CDS Authorizations. .......................................................... 10
      Default BOPF Authorizations. ........................................................ 10
   5.3 Generic Authorization Provider for Authority Check. ...................... 11
   5.4 Data-Aware Authorization Provider for Authority-Check. ............... 12
   5.5 Application-Specific Authorization Providers for Authority Check. .... 13
   5.6 Applications Without Authorization Protection. ............................ 15
   5.7 Applications Not Using the ABAP Authority Check. ....................... 15
   5.8 Testing with Authorization Providers. ............................................... 16
6. **Appendix.** ................................................................................................. 17
   6.1 Terms. .................................................................................................. 17
   6.2 Support and Issue Reporting. .............................................................. 17
1 Authorizations for SADL-Based Applications

This document provides a detailed guide on how to use the SAP authorizations in your SADL-based applications.

In this guide you will learn how to:

1. Analyzing your authorization requirements and deciding the best authorization provider to use
2. Using default authorization providers
3. Using generic authorization providers
4. Providing data-aware authorizations*
5. Providing applications without authorization enforcement
6. Creating your own authorization providers
7. Testing applications with authorization providers

Note

* Only available with SAP NetWeaver 7.4 SP07 and higher.
2 Document History

Table 1:

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4 SP04 V1</td>
<td>First official release of this guide, valid for SAP NetWeaver 7.4 SP04</td>
</tr>
<tr>
<td>7.4 SP07 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP07</td>
</tr>
<tr>
<td>7.4 SP10 V1</td>
<td>Version valid for SAP NetWeaver 7.4 SP10</td>
</tr>
</tbody>
</table>
3 Prerequisites

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

- The user must have a reasonable knowledge of ABAP authorizations.
- SADL integration is provided for SAP NetWeaver 7.4 SP04 and higher.
- Some of the features are available with support package level of SAP NetWeaver 7.4 SP07 and higher.
4 Background Information

4.1 Authorization Enforcement

Authorization enforcement consists of three major activities:

1. Definition of the authorization-relevant attributes (for example, authorization object fields): The application developer defines the fields that are relevant for an authority check (for example, the type of a business document).
2. Configuration of the end user authorizations (for example, by roles): An administrator configures the user-specific values related to authorization-relevant fields (for example, which product types can be edited by a certain end user).
3. Authorization check (for example, in ABAP authority check): The runtime working with data has to match the user authorizations against the requested access to business data in order to restrict data to the allowed data subset.

4.1.1 Authority Check in ABAP

In ABAP the built-in authority check function indicates whether a given data record matches the end user authorization or not. Since there is no modeling information linking authorization objects to DDIC tables in ABAP, the application has to define the mapping in the coding. At runtime the application calls the authority check for each record and maps the authorization object fields of the read data onto the allowed user parameters.

4.1.2 Other Authorization Mechanisms

In addition to the ABAP authority check, there are other authorization concepts - for example, OAuth tokens, RBAM ACLs, or HANA privileges. While the implementation and the manifestation of the user authorizations differ, the idea behind any authorization concept is to compare some attribute of the user versus an attribute of the data the user is accessing. The point in time for doing this is generally after retrieving the data, such as during the ABAP authority check, in order to be able to compare the attributes with the user data.
### 4.1.3 Authority Check in Query Push-down

SADL Query enables fast read access for scenarios on mobile and desktop applications based on HANA by means of a query push-down. As part of the query push-down, all users’ input is collected through consumer APIs and used to configure the request for the database. Specifically, the authorization enforcement in this process is interposed between query specification by application or end user and data retrieval from the database. As a result, only datasets for which the user is authorized are requested from the database, reducing both the load on the database, the traffic between database and application, and the amount of calls to the authorization enforcement engine.

The basic idea is that the application provides the parameters for the authority-check. Before data retrieval is executed, the following steps are performed by the Query Engine:

1. Evaluation of authorization of the active user
2. Mapping of authorization object fields to the table / view columns or business entity attributes, or OData service properties, respectively.
3. Adding authorization restrictions to the condition for the database select (\texttt{WHERE}-clause).

### 4.1.4 Authority Check in Transactional Scenarios

Transactional services using SADL entities delegate the authorization enforcement to the framework performing the actual modifications.

### 4.2 Authorization Condition Providers

SADL supports the concept of authorization condition providers. The providers are objects responsible for reading the business entity metadata, evaluating the authorizations of the active users, and transforming the result into suitable restriction information for the database selection. With respect to the provision of authorization information, we distinguish between business entities with built-in authorization information (referred to as modeled authorizations), and business entities that are agnostic of authorizations, and rely on an outside check before access is given (also referred to in the following as application-defined authorizations).

#### 4.2.1 Modeled Authorizations

Modeling information defines the authorization-relevant fields (for example, by ACL paths in BOPF) so that SADL can generically determine which database fields are relevant for the restrictions for the query push-down. The applications built on business entities with modeled authorizations do not have to provide additional code or data for the authority check. They may take advantage of the built-in authorization provider for their respective SADL business entity type.
4.2.2 Application-Defined Authorizations

If the situation arises where no modeling information is available (for example, DDIC), SADL cannot generically determine which authorization it should check for a specific business entity. Here, the application has to provide the information for each usage scenario of a business entity. The applications can use generic or their own condition providers in order to define the missing authorization-relevant metadata for a business entity or a business scenario.
5  Step-by-Step Procedure

5.1  Analysis of Authorization Requirements

The prerequisite for deciding on using one of the following authorization providers, or creating a new one, is the analysis of the protection requirements of an application.

Context

This decision can be helped by answering the questions below:

1. Does the application need authorization protection? If not, use the provider for no authorization check.
2. Does the application use the ABAP authority check? If not, create your own authorization provider.
3. Is authorization metadata for the business entities used in the application and does it match the authorization attributes required by the application? If the answer is ‘yes’, use the default authorization provider. If not, use the parameterized generic authorization provider.

5.2  Default Authorization Providers

Default authorization providers belong to business entities with authorization metadata (for example, a business entity of CDS or BOPF type).

Context

They rely on the model information for deciding on the required authorizations and the attributes that are authorization-relevant.

If the modeled authorizations are sufficient for the application, no further action must be taken. The default authorization provider is instantiated transparently and used by the SADL Engine.

If they are not sufficient, the application may use a parameterized generic authorization provider (if applicable), or create and provide its own authorization provider, as described in the following paragraphs.

Authorization providers created and registered by the application will always overwrite default authorization providers.
5.2.1 Default CDS Authorizations

Default CDS authorizations are modeled as Data Control Language (DCL) roles. There can be multiple DCL roles for each CDS entity. The roles enable access to a CDS entity:

- Partially, to access only some instances of the CDS entity, depending on your authorization
- Completely, to access all instances of the CDS entity, provided you have been assigned the respective role

Mapping roles are a special category of DCL roles that map elements of the CDS entity onto fields of authorization objects. These mapping roles are not assigned to a user, but will grant you access to the CDS entity – based on your assigned authorization objects contained in your classical roles.

**Note**

This feature is available since SAP NetWeaver 7.4 SP10.

5.2.2 Default BOPF Authorizations

BOPF has a generic authorization concept based on authorization objects and the authority-check statement. The authorization objects must follow a specific pattern and are assigned to business object nodes. It is possible to assign multiple authorization objects to one node. In this case, every object will be checked at runtime.

Similar to the Core Data Services (CDS) default authorizations, the BOPF authorizations will allow access to a business object (BO):

- Partially, instance-based, if authorization fields are mapped to node elements
- Completely, if no authorization field mapping is done.

All BO services are protected by the default authorization check. Even if no element mapping is done, at least the called service is checked against the user authorization at runtime.
5.3  Generic Authorization Provider for Authority Check

SADL provides a generic authorization provider that can be parameterized by the application in order to specify the required authority check parameters. This condition provider can be used if the application makes use of the ABAP authority check concept, but the used business entity does not specify the authorization metadata.

Context

The SADL infrastructure provides the generic authorization provider class `CL_SADL_COND_PROV_AUTH_OBJECTS` for handling ABAP authority checks. This provider must be initialized before passing control to the SADL engine for query execution.

In order to make use of this class, perform the following steps:

Procedure

1. Get an instance of this class using the condition provider factory.

   ```
   DATA(lo_provider) = 
   cl_sadl_cond_prov_factory_pub=>create_for_authorization( ).
   ```

2. Use the method `add_authorization_for_object()` for specifying the authorization objects to be checked.

   ```
   lo_provider->add_authorization_for_object( 
   iv_authorization_object = 'S_EPM_BP' 
   it_activities = VALUE #( ( auth_field = 'ACTVT' value = '03' ) ) 
   it_field_mapping = VALUE #( ( auth_field = 'EPM_BP_ID' view_field = 'MY_BP_ID' ) ) ).
   ```

   The `add_authorization_for_object()` method takes as input exactly one authorization object, a list of activity-like fields and their values that denote read authorizations, and a list of mappings between fields of the authorization object and attributes of the business entity; in this example, mappings between authorization fields and columns of the database view.

3. If you need to check more than one authorization object, you can call the method `add_authorization_for_object()` multiple times.

   In this case, the resulting restrictions will be applied sequentially to the data selection (equivalent to an 'AND' between authorization objects). The result is that only the data that passes all authority checks will be returned.
5.4 Data-Aware Authorization Provider for Authority-Check

Data-aware authorization providers are able to perform different checks depending on the content of the data. For example, perform an authorization check only if a specified data field is not initial, or uses different authorization objects for different values in a database column.

Prerequisites

This feature is available since SAP NetWeaver 7.4 SP07.

Context

SADL provides a generic authorization provider that is data aware. This condition provider can be used if the application makes use of the ABAP authority check concept, but the used business entity does not specify the authorization metadata, and in addition the business requirements are such that the authorization check needs to adapt to the actual data content, as described previously.

This provider must be initialized before passing the control to the SADL engine for query execution.

In order to make use of this class, perform the following steps:

Procedure

1. Get an instance of the data-aware condition provider.

   ```
   DATA(lo_provider) = cl_sadl_cond_prov_factory_pub=>create_for_constr_auths( ).
   ```

2. Use the method `add_authorization_conditions()` for specifying the authorization objects to be checked.

   For each authorization object, you should provide the conditions under which the check is to be performed. You can provide the condition as a table of named ranges using the constraints parameter.

   ```
   lo_provider->add_authorization_conditions( VALUE #( 
     ( authorization_object = 'S_EPM_BP' 
       activities = VALUE #( ( auth_field = 'ACTVT' value = '03' ) ) 
       field_mapping = VALUE #( ( auth_field = 'EPM_BP_ID' view_field = 'MY_EPM_BP_ID' ) ) 
     ) 
     constraints = VALUE #( ( name = 'BP_ROLE' range = VALUE #( ( option = 'EQ' low = '01' ) ) ) )
   ).
   ```

   The `add_authorization_conditions` method takes as input a table of authorization objects with the respective activities, field mappings and constraints.

3. Leaving the constraints empty has the result that the check is always executed. In this case, the data-aware condition provider behaves exactly like the generic authorization provider.
4. Leaving the authorization object empty has the result that for the specified constraints no check is performed. In the following example, the authorization object `S_EPM_BP` is checked only of the database field `BP_ROLE` contains a non-initial value.

```plaintext
lo_provider->add_authorization_conditions( VALUE #(
    { authorization_object = 'S_EPM_BP'
    constraints = VALUE #(
        { name = 'BP_ROLE' range = VALUE #(
            { option = 'EQ' low = '' } )
    )
    )
).
```

5. If you need to check more than one authorization object, you can call the method `add_authorization_conditions` multiple times.

In this case, the resulting restrictions will be applied sequentially to the data selection (equivalent to an 'AND' between authorization objects). The result is that only the data that passes all authority checks will be returned.

6. If you need to check multiple authorization objects alternatively, depending on different values of the constraint, you will need to place the authorization objects into the same call to the `add_authorization_conditions` method. This is equivalent to an OR between authorization objects, whereby each check will only be performed if the constraints allow it.

In the following example, the authorization object `S_EPM_BP` is checked for role '01' (Customer), and the authorization object `S_EPM_PO` is checked for role '02' (Supplier).

```plaintext
lo_provider->add_authorization_conditions( VALUE #(
    { authorization_object = 'S_EPM_BP'
    activities           = VALUE #( ( auth_field = 'ACTVT' value = '03' ) )
    constraints          = VALUE #(
        { name = 'MY_BP_ROLE' range = VALUE #(
            { option = 'EQ' low = '01' } )
    )
    )
    { authorization_object = 'S_EPM_PO'
    activities           = VALUE #( ( auth_field = 'ACTVT' value = '03' ) )
    constraints          = VALUE #(
        { name = 'MY_BP_ROLE' range = VALUE #(
            { option = 'EQ' low = '02' } )
    )
    )
).
```

⚠️ Caution

The data returned to the caller will contain only business partners where the field `BP_ROLE` contains '01' nor '02'. In order to add this data to the result set without checking authorizations, you need to explicitly add the authorization conditions with the respective constraints (`BP_ROLE = '01'`) and without an authorization object, as described under topic 4.

5.5 Application-Specific Authorization Providers for Authority Check

Application-specific authorization providers should be used in the case where neither the default authorization providers of the business entity nor the generic authorization provider are suitable for the purpose of the application. The reasons for the latter are either that reuse of the authorization metadata is intended for multiple calls or that the application does not make use of the ABAP authority check, but has a different authorization concept.
Applications that use the authority check can create an individual authority provider by subclassing a
convenience class cl_sadl_cond_prov_authobj_base. This provider must be initialized and registered
with the SADL runtime before passing the control to the SADL engine for query execution.

**Note**

This is the same information provided directly to the generic condition provider in the previous chapter. The
default authorization providers receive this kind of information from the business entity.

Use this approach if you:

- Plan to reuse the authorization provider in multiple applications, with the same authorization
  parameters
- Create a generic implementation
- Provide an authority provider for an FPM Application. Since FPM directly instantiates the authorization
  provider, there is no consumer code available to set the authority check parameters for the generic
  provider; thus an application specific provider must be used.

To do this, perform the following steps:

**Procedure**

1. Create your own authorization provider class that inherits from
   the cl_sadl_cond_prov_authobj_base.

   ```
   METHOD constructor.
   super->constructor( ).
   add_authorization_for_object(
     iv_authorization_object = 'S_EPM_BP'
     it_activities = VALUE #( ( auth_field = 'ACTVT' value = '03' ) )
     it_field_mapping = VALUE #( ( auth_field = 'EPM_BP_ID' view_field = 'MY_BP_ID' ) )
   )
   ENDMETHOD.
   ```

2. Create a constructor method in this class. This constructor can be used to set the authorization metadata
   for the authorization provider.

   See the previous chapter for the meaning of the parameters.
5.6 Applications Without Authorization Protection

Context

Applications that have no need of an authorization check because they are publicly available must explicitly declare that no authorization check is required. To indicate this, use the special authority provider that always allows access:

```
DATA( lo_provider ) = cl_sadl_cond_prov_factory_pub=>
create_for_no_authority_check ( ).
```

This provider must be initialized and registered at the SADL runtime before passing control to the SADL engine for query execution.

⚠️ Caution

All access to business data through this application will be allowed, regardless of the user authorizations. In the case where the application does not provide an authorization provider, the default authorization provider is used by the SADL Engine. This means that for DDIC the table display authorization is checked (as in transaction SE16).

5.7 Applications Not Using the ABAP Authority Check

Context

Applications that follow another authorization concept should create their own authorization provider. In this case, perform the following steps:

Procedure

1. Create an application provider class that implements the interface `if_sadl_cond_prov_auths`.
2. Create an implementation for the two required methods of this interface `is_authorized` will return true if the authorization checks performed have established that the user is authorized to access at least part of the requested data. `get_conditions` will return a list of restrictions in the reverse polish notation defined in `if_sadl_query_types`.
5.8  Testing with Authorization Providers

Context

The application tests are best performed with adequate test users and the matching authorizations. Since this is not always possible, SADL provides some test helpers that simulate authorizations for the current user. These test helpers can only be used in test classes.

To be able to use them, perform the following steps:

Procedure

1. Create an instance of the condition provider.
   
   ```
   DATA( lo_provider ) = NEW appl_cond_provider ( ).
   ```

2. Use the SADL test helper to inject the desired user authorizations. This can only be done within test methods.
   
   ```
   cl_sadl_consumer_aunit_friend => inject_auth_object_values ( 
     io_provider = lo_provider 
     it_auth_values = VALUE #( ( auth = 'S_ROLE' 
     objct = 'S_EPM_BP' field = 'EPM_BP_ID' von = '1' bis = '10' ) )
   )
   ```

3. Execute your test and verify the result. For example, verify that the service under test returns less result lines for restricted authorizations. The method used in this example simply checks that the generated condition matches expectations.
   
   ```
   lo_provider => get_condition ( 
   EXPORTING it_elements = VALUE #( 
   ( id = 'MY_BP_ID' column = 'BP_ID' table = 'SNWD_BPA' ) ) 
   IMPORTING et_sadl_condition = DATA( lt_condition ) ).
   cl_abap_unit_assert => assert_equals ( 
   act = lo_provider => is_authorized ( ) 
   exp = abap_true ).
   cl_abap_unit_assert => assert_equals ( 
   act = lt_condition 
   exp = VALUE if_sadl_query_engine_types => tt_complex_condition ( 
   ( type = 'simpleValue' value = '1' ) 
   ( type = 'simpleValue' value = '10' ) 
   ( type = 'between' attribute = 'MY_BP_ID' ) 
   ( type = 'notIsNull' attribute = 'MY_BP_ID' ) 
   ( type = 'and' ) ) ).
   ```

4. Remove injected user authorizations.
   
   ```
   cl_sadl_consumer_aunit_friend => reset_auth_object_values ( ).
   ```
6 Appendix

6.1 Terms

Table 2:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
</table>
| Business Entity                      | • Entities that are used in business applications – for example, business objects, database tables, search views.  
• 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. |
| Business Entity type (or source)     | Specific repository or provider for business entity metadata – for example, Business Object Processing Framework, Data Dictionary, Core Data Services |
| SADL Engine                          | Code performing the query execution, including view-building based on business entities.                                                      |

6.2 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).