Developing roles in SAP HANA
Applicable to SAP HANA releases starting from 1.0 SPS12 and newer

Document version 1.2 2021-07
Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Release Date</th>
<th>Change description</th>
<th>Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>April 2018</td>
<td>Document creation</td>
<td><a href="mailto:askSAPHANA@sap.com">askSAPHANA@sap.com</a></td>
</tr>
<tr>
<td>1.1</td>
<td>November 2018</td>
<td>Minor updates: latest recommendations, re-wording of some paragraphs, footnotes and correction of typos.</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>July 2021</td>
<td>Updates: Split, news, recommendations, paragraphs, figures, correction of typos, links and layout (design)</td>
<td></td>
</tr>
</tbody>
</table>

References

General
- **SAP HANA Developer Guide:** Explains how to build applications using SAP HANA, including how to model data, how to write procedures, and how to build application logic in SAP HANA Extended Application Services, classic model.
- **SAP HANA XSA Developer Guide:** Explains how to build applications using SAP HANA, including how to model persistent and analytic data, how to write procedures, and how to build application logic in SAP HANA Extended Application Services advanced model (XS advanced).
- **SAP HANA Security Guide:** Is the entry point for all information relating to the secure operation and configuration of SAP HANA.
- **SAP HANA Security Checklist:** Offers recommendations and information about optimizing your security configuration to help you run your SAP HANA securely.
- **SAP Note 2465027:** Deprecation of SAP HANA extended application services, classic model and SAP HANA Repository.

Migration
- **SAP HANA XS Advanced Migration Guide:** Explains how to prepare, migrate, and deploy applications from the XS classic model to the XS advanced model.
## Glossary

Following abbreviations will be used throughout the document:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>DU</td>
<td>Delivery unit</td>
</tr>
<tr>
<td>HDB</td>
<td>SAP HANA database</td>
</tr>
<tr>
<td>HDI</td>
<td>SAP HANA deployment infrastructure</td>
</tr>
<tr>
<td>LCM</td>
<td>SAP HANA lifecycle management</td>
</tr>
<tr>
<td>UPS</td>
<td>User provided service</td>
</tr>
<tr>
<td>XSA</td>
<td>SAP HANA extended application services, advanced model</td>
</tr>
<tr>
<td>XSC</td>
<td>SAP HANA extended application services, classic model</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS

Document history ........................................................................................................ 2  
References .................................................................................................................... 2  
Glossary ......................................................................................................................... 3  
1. **INTRODUCTION** ............................................................................................... 6  
2. **PREPARATION** ............................................................................................... 7  
   2.1 Prerequisites .................................................................................................. 7  
   2.2 SAP HANA release ....................................................................................... 7  
3. **WHAT IS CHANGED FROM XSC TO XSA?** .................................................. 8  
   3.1 Development Scenarios .............................................................................. 8  
   3.1.1 SAP HANA and XSC .............................................................................. 8  
   3.1.2 SAP HANA and XSA ............................................................................. 9  
   3.1.3 Orgs, spaces and projects .................................................................... 10  
3.2 Database roles .................................................................................................... 11  
   3.2.1 Design-time roles created in XSC .......................................................... 11  
   3.2.2 Design-time roles created in XSA .......................................................... 11  
   3.2.3 Comparing design-time roles between XSC and XSA ....................... 12  
4. **HDI** ................................................................................................................... 14  
   4.1 Container model .......................................................................................... 14  
   4.2 Container groups .......................................................................................... 14  
   4.3 Container isolation ...................................................................................... 15  
   4.4 Breaking the container isolation ................................................................. 16  
5. **ROLE DEVELOPMENT IN XSA** .................................................................. 17  
   5.1 The steps before development can start ..................................................... 17  
   5.2 Equip the HDI container with external privileges ....................................... 17  
   5.2.1 Between HDI containers in the same space ........................................... 18  
   5.2.2 Between HDI containers in different spaces and external objects ....... 18  
   5.2.2.1 Using a UPS with a procedure grantor ............................................... 19  
   5.2.2.2 GRANT to #OO user directly ............................................................. 20  
5.3 Role development process .................................................................................. 20  
6. **BEST PRACTICES AND RECOMMENDATIONS** ......................................... 21  
   6.1 Organization and spaces .............................................................................. 21  
   6.2 Setup ............................................................................................................. 21  
   6.3 Other considerations and hints .................................................................... 22  
7. **MIGRATING ROLES FROM XSC TO XSA** ................................................... 23
1. **INTRODUCTION**

The objective of this document is to describe the best practices and general recommendations on the role development process for SAP HANA database (HDB) access roles with SAP HANA and SAP HANA Extended application services, advanced model (XSA).

XSA provides a comprehensive platform for the development and execution of native data-intensive applications that run efficiently in SAP HANA. It thus succeeds XS classic (XSC) as the default application programming model for SAP HANA. This implies that XSA defined and deployed roles will substitute the XSC defined and activated design-time roles.

The purpose of this document is not to replace the SAP HANA XSA Developer Guide, but to provide additional information and recommendations on how to address the common challenges on the role development process with XSA.

Additionally, in this document, we deliver and describe SAP HANA Deployment Infrastructure (HDI)-based role templates for administrative tasks in SAP HANA database. This HDI-based role templates can be used by costumers as a starting point for the creation of their own roles.

Please note that this document does not cover user management of XSA applications themselves currently!
2. PREPARATION

2.1 Prerequisites
This document aims at administrators and role developers who:

- are experienced with SAP HANA,
- are familiar with SAP HANA privileges, roles as well as XSC and
- have basic knowledge about XSA (e.g. have attended the latest openSAP course on SAP HANA development).

2.2 SAP HANA release
SAP HANA XSA is shipped as an optional component for the following releases of SAP HANA, so this document only applies in case we have one of the following releases installed or are planning to move or upgrade to one of the following SAP HANA releases:

- SAP HANA 1.0 SPS12,
- SAP HANA 2.0 SPS00 or higher.

Furthermore, SAP note 2347931 explains the relation between SAP HANA and release versions of the SAP HANA XS, advanced model components.
What is changed from XSC to XSA?

3. WHAT IS CHANGED FROM XSC TO XSA?

Since a lot has changed in the transition from XSC to XSA, the most important changes in terms of role development are discussed below. Please also refer to the developer guide and the XSA guides mentioned in section “References”.

3.1 Development Scenarios

3.1.1 SAP HANA and XSC

Previously, with SAP HANA and XSC, the development of HDB artifacts was done through SAP HANA Studio or with the SAP HANA Web-based Development Workbench. So, a developer using one of those tools creates a design-time version of an object, such as a view, a procedure or a role and this design-time version is saved within the SAP HANA Repository. By activating the design-time object, a runtime version of the object (also known as catalog object) is created by the _SYS_REPO technical user. In this setting, there is only one repository known as SAP HANA repository and it is owned by the _SYS_REPO user. All objects generated from the SAP HANA repository are also owned by the _SYS_REPO user.
3.1.2 SAP HANA and XSA

With the arrival of XSA, it has changed the way we develop HDB objects compared to XSC. Developers will use the SAP Web IDE in XSA to create the design-time version of an HDB object, such as a view, procedure or role. The design-time objects are created within a project can be stored in a version control system like a GIT repository.

When the developer deploys the project, a runtime version of the objects is created in an HDI container within the HDB.

An HDI container can be seen as a database schema and we can have multiple HDI containers within the HDB. All database objects deployed within the container are owned by container-specific technical object owner.
3.1.3 Orgs, spaces and projects

In XSC all the development objects were stored in the SAP HANA repository which was composed by a hierarchy of packages. There were no granular admin privileges and we could use “package privileges” to allow developers to create, edit or activate objects within a specific package and its sub-packages.

With XSA, a new controller model has been introduced which is composed of a hierarchy of organizations and spaces. This new model allows us to separate administration privileges per organization and space and to isolate resources and privileges between applications that are in different spaces.

The controller model of XSA introduces the following terms:

- **Organization**: can be used e.g. to logically separate customers or development areas. One initial organization is always created during the installation. Organizations can be used to separate:
  - line of business development,
  - security-related development from other development and
  - sets of privileges for role-building.

- **Space**: an organization can contain one or more spaces. Spaces can be used to physically separate applications. Services are always available for all applications in the same space. The first space created during installation is named SAP and contains all services needed to develop and deploy applications with XSA. Spaces can be used to separate:
  - security-related development from other development
  - sets of privileges for role-building

- **Application**: in a space, one or more applications can be deployed. They share the services of their space.

- **Projects**: can be used to separate e.g. sets of roles for different target systems. A developer can create multiple projects within a space.
3.2 Database roles

In SAP HANA we have two ways of creating database roles:

- directly at run-time (known as catalog roles)
- using a role definition (known as design-time roles).

SAP recommends working with design-time roles and due to XSA and the new HDI concept, there is a significant difference in the way they are created and managed compared to the design-time roles in XSC – refer to section “Comparing design-time roles between XSC and XSA”.

3.2.1 Design-time roles created in XSC

In XSC, the runtime versions of the design-time roles are global in the database. As there is a single repository in the database, all the design-time roles are owned by the _SYS_REPO user and can only be equipped with privileges that are granted to the _SYS_REPO user with grant/admin option. This means that the _SYS_REPO user holds almost all privileges on the system.

Granting or revoking a design-time role is only done via database procedures owned by the _SYS_REPO user - e.g. CALL_GRANT_ACTIVATED_ROLE. These roles cannot be granted by a user with ROLE ADMIN privilege.

Additionally, application development and role development are mixed up. As there is only one repository that it is owned by the _SYS_REPO user and developers can create different type of objects, application developers can take advantage of this setup to create roles with wider privileges for their applications.

3.2.2 Design-time roles created in XSA

On the other hand, in XSA, runtime versions of the design-time roles are created at schema-level due to containerization. These roles are owned by the container-specific technical user and can be compound by privileges granted with admin/grantable option to the container-specific technical owner.

Due to this containerization, it is possible to separate role development from the application development as this can be organized within different spaces in XSA and thereby possible to only assign the relevant privileges to the respective container-specific technical users.

To grant or revoke a role created with XSA there are two possibilities:

- Using a database procedure which is provided as part of HDI container API by default.
- Using the system privilege ROLE ADMIN that allows to grant/revoke roles system wide.
What is changed from XSC to XSA?

3.2.3 Comparing design-time roles between XSC and XSA

As mentioned before, there are many differences between roles developed on XSC (repository roles) and the roles developed on XSA (HDI-based roles). The principal differences are:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Design-time repository roles (XSC)</th>
<th>Design-time HDI roles (XSA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transportability</td>
<td>• SAP HANA Application Lifecycle Manager</td>
<td>CTS+ of the SAP NetWeaver ABAP application server</td>
</tr>
<tr>
<td></td>
<td>• The change and transport system (CTS+) of the SAP NetWeaver ABAP application server</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• SAP HANA Transport Container (HTC)</td>
<td></td>
</tr>
<tr>
<td>Version management</td>
<td>The repository</td>
<td>GIT repository</td>
</tr>
<tr>
<td>Ownership</td>
<td>_SYS_REPO</td>
<td>Technical user &lt;container where role development is taking place&gt;#OO)</td>
</tr>
<tr>
<td>Grant and revoke process</td>
<td>Built-in procedures</td>
<td>• HDI container’s API schema or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Container group’s API schema or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ROLE ADMIN</td>
</tr>
</tbody>
</table>

Refer to section “Catalog Roles and Design-Time Roles Compared” of the SAP HANA Security Guide for SAP HANA Platform for more info.

In the following diagram, we can see the differences between roles (design-time and run-time roles) in XSC and in XSA.
What is changed from XSC to XSA?

There are also differences in the way a role can be granted depending if they are global roles (XSC) or schema-local roles (XSA). Global roles cannot be granted or revoked with ROLE ADMIN system privilege, but through the SYS_REPO API, thus the execute privilege on the procedures is required.

Conversely, schema-local roles can be granted or revoked with ROLE ADMIN system privilege and/or through the container-local API. This characteristic allows us to grant all the roles in the system by having ROLE ADMIN system privilege or only grant roles that are contained in a specific container where we have the execute privilege on container-local API.
4. HDI

With XSA it was also introduced the HDI which is a service layer of the HDB that simplifies the deployment of HDB artifacts.

4.1 Container model

As discussed previously, all HDB artifacts, such as roles or tables, that are created through XSA are deployed in the HDB within an HDI container.

An HDI container is essentially a database schema. Each container has its own container object owner (also known as #OO user), some API and a schema for storage.

4.2 Container groups

HDI containers can be group into HDI container groups for administrative purposes. An HDI container administrator can create container groups, assign containers to a container group and manage container group administrative privileges. A container group role is created per container group. This role is granted by default to all the container object owners that belong to the same container group. Container group roles were released with SAP HANA 2.0 SPS03.
4.3 Container isolation

HDI containers provide security isolation as it is based on a “zero” privileges principle by default. All objects within a container are owned by the #OO user. This object owner has the following characteristics:

- is a restricted database user,
- has CREATE ANY privileges on the schema and
- has no external privileges by default.

By default, references to external objects are not allowed. For example, view X has no direct reference to table ERP.Y:

The HDI container isolation also affects the creation of database roles through XSA. When we create a database role, using XSA, the database role will be deployed within an HDI container and it will be owned by the database object owner. Thus, to be able to deploy the roles, the role can only contain privileges that are assigned to the #OO user with GRANT / ADMIN option. Otherwise, the deployment will fail due to a “Missing authorization error”.
4.4 Breaking the container isolation

To break the HDI container isolation, we need to grant the privilege (with GRANT/ADMIN option) of the external object to the #OO user.

In the previous example, to create a view X as table ERP.Y, the following actions need to be performed in advance:

- Grant SELECT privilege on ERP.Y to the #OO user with GRANT/ADMIN option.
- Create a synonym Y that points to table ERP.Y

In case of database roles, the approach is similar. To deploy successfully a database role, all the privileges that are included in the role should be granted to the #OO user with GRANT / ADMIN option in advance.

There are different alternative options how to grant the database-level privileges to the #OO user. These options are described in the following section “Equip the HDI container with external privileges”.

5. ROLE DEVELOPMENT IN XSA

For the development of roles, no real application is created, only database artifacts – the roles – are used. For this, developers will use SAP Web IDE to work on a project within a space in XSA where they will create the design-time objects. In the space, only two types of services are needed to create roles:

- HDI container and
- Optional: User provided service (UPS).

In this section, we are going to describe the steps needed to develop transportable HDB roles using XSA.

5.1 The steps before development can start

The initial step is to install and prepare XSA and SAP Web IDE application. Once the setup is ready, we need to arrange our organization and spaces in a way that suits the needs of our organization.

How to create spaces, setup users and call the SAP Web IDE can be found in chapter “Post-Installation Administration Tasks” in the SAP Web IDE for SAP HANA - Installation and Upgrade Guide.

It is recommended to setup at least a dedicated space for role developing – refer to section “Best practices and recommendations”.

5.2 Equip the HDI container with external privileges

In the following section are described the different alternatives to grant database-level privileges to the container object owners. These privileges can be:

- System privileges,
- Object privileges and
- Analytic privileges.

For illustration purposes, assume to deploy the database roles within the HDI container named “C1” and grant all the external privileges to the container object owner user (C1#OO user) with GRANT /ADMIN option in advance.
5.2.1 *Between HDI containers in the same space*

When two or more containers are in the same space, every HDI container is shared (accessible) to other HDI containers in the same space. This is possible as it exists a so-called trusted setup between all the resources located in the same space. In this case, there is a shortcut mechanism where the C1 container itself can obtain privileges for the other container in the same space, e.g. C2.

5.2.2 *Between HDI containers in different spaces and external objects*

In the case of an external schema (X schema) or an HDI container living in a different space, it is more complex since the owner of the object needs to explicitly grant the privilege. The same applies to system privileges or another type of objects such as tables or views.
The following two options are recommended to grant the privileges of an external object to the C1#OO user:

5.2.2.1 Using a UPS with a procedure grantor

At HDB level, we will need to setup two different user accounts:

- User A: This is a standard user that have all the required privileges with GRANT/ADMIN option. This user owns a database procedure running in definer mode that will be used to grant the privileges to the #OO user. This user account can be deactivated and there is no need to share its credentials.
- User B: This is a standard user account with privileges to EXECUTE the database procedure owned by the user A. This user needs to be activated and the credentials need to be shared as it will be exposed to the XSA space.

If user A will be used for building roles only containing privileges already granted to SYSTEM by default (e.g. SYSTEM PRIVILEGES), the SYSTEM user could be used as user A. If user A will be used for building roles containing privileges of e.g. developed objects, a dedicated user should be created and used as user A instead of SYSTEM user.

Additionally, in the XSA space, we need to define a UPS with the user B credentials. This UPS will be used by the HDI deployer (application used to “prepare” the HDI container) to provide automatically the required privileges to the #OO user.

To define which external privileges are going to be granted to the #OO user, we need to maintain in the XSA project the .hdbgrants file, specifying the UPS as granting service.

On startup, the HDI deployer will process the .hdbgrants file, connect to the database using the UPS credentials (user B credentials), and execute the procedure that grants the privileges to the #OO user.

Again, do not use the user SYSTEM as the user A, except for role-building with privileges granted to the SYSTEM user by default - e.g. SYSTEM PRIVILEGES.
Role development in XSA

The user account that has all the required privileges (user A) can be deactivated and the credentials are not shared. Instead, the credentials of the user B, with very limited privileges, are shared to define the UPS.

Additionally, further security validations can be implemented within the procedure’s logic to avoid granting the privileges to a different user than container object owners. For example, validating that the grantee user is deactivated and the username ends with “#OO” characters.

The procedure grantor mechanism is supported as of version 3.4.1 of the @sap/hdi-deploy component in XSA.

5.2.2.2 GRANT to #OO user directly

This option consists of granting the external privilege or role directly to the #OO user. After this, we will be able to deploy the database role within the HDI container.

This approach is like the one used with XSC when we needed to grant to the _SYS_REPO user the privilege with GRANT / ADMIN option to include it in a design-time role.

When using this option, the following challenges exist:

- The #OO is automatically created during the “build” process when the HDI container is also created. This will be an issue when transporting the roles as the first time when we build and deploy the project, as #OO will be created with minimum privileges and we will get a “missing authorizations” error.
- In a development scenario, we can potentially have more than one container for role building (one per developer) so we will need to manually grant the privilege or role to each #OO user.

5.3 Role development process

Role development is done by the developers in the SAP Web IDE. There the developer will create a new project in the space created for the roles. The developer creates an HDB module which automatically deploys to an HDI container on the SAP HANA DB. The services needed for role developing needs to be made known to the project. This is done in the mta.yaml file. After this, the developer can create now:

- file for granting privileges via the UPS - .hdbgrants,
- synonyms for object privileges - .hdbsynonyms,
- role definitions - .hdbrole and
- (perhaps) role configuration files - .hdbroleconfig.

For further details on how to create the different development objects needed for role developing – refer to SAP HANA Developer Guide for SAP HANA XS Advanced Model.
6. BEST PRACTICES AND RECOMMENDATIONS

6.1 Organization and spaces

For role development, there is no unique setup of organizations and spaces that are ideal for this purpose. The possibility to organize privileges and roles into organization and spaces will give us the flexibility to cover all the different scenarios we could find.

We should take into consideration that, while more spaces and organizations in the XSA environment, the complexity also increases. Therefore, keep it as simple as possible.

As a principle, clearly differentiate security-related development, like role development, from application development. Application developers will need to create roles to access their developed artifacts, but those roles should only include privileges related those artifacts and not privileges for database administration for example.

This is avoidable by having a dedicated organization or space within the organization for safety-related development.

Another recommendation is to separate into different organizations or spaces the roles for database administration from the roles for a specific functional area, for example, roles for HR applications.

6.2 Setup

It is recommended to use the option "Using a UPS with a procedure grantor" in all environments as a mechanism to grant the external privileges to the #OO user of the container where the database roles are being deployed.

With this mechanism, no activated user needs to have all the granted privileges. Only the call of the procedure is exposed, and the procedure can contain various additional security checks.

To avoid delays in the development process due to missing authorizations in the container, depending on your scenario the following recommendations apply when developing roles:

- only containing privileges already granted to SYSTEM by default (e.g. SYSTEM PRIVILEGES), the SYSTEM user should be used as the user holding all the privileges (User A as explained in section “Using a UPS with a procedure grantor”).

In this case, the space where the UPS is exposed should be used exclusively for role development and not for application development.

- containing privileges of e.g. developed objects, it is recommended to grant wide privileges to the database user holding all the privileges.

Afterwards, developers can assign needed privileges to the HDI container through the grant file definition (.hdbgrants).

For a production environment, depending on your scenario the following recommendations apply when developing roles:

- containing privileges of e.g. developed objects, it is recommended to grant to the database user (user A in section “Using a UPS with a procedure grantor”) only the privileges required to deploy the project. The list of required privilege can be obtained from the grant file definition (.hdbgrants) in the DEV environment.
6.3 Other considerations and hints

- Use SYSTEM user to initially set up the user providing the privileges.
- Be aware, that a role developer for HDB roles will have critical privileges if we allow him to test the roles in the database.
- Segregate different sets of privileges in different spaces or even different organizations.
- Separate sets of roles meant for different target systems in different projects - e.g. test and productive system.
- Do not mix XSC and XSA role development - e.g. do not include XSA built roles in XSC defined roles.
- Everything discussed in this chapter holds for all databases in your system, i.e. all tenants and the SYSTEMDB. Technically this is solved by mapping different spaces with different databases.
7. **MIGRATING ROLES FROM XSC TO XSA**

For the migration, we assume that you currently develop your roles in XSC, and you have an existing role concept for SAP HANA for development and administration tasks. Now you want to make the switch from XSC to XSA and therefore you also must migrate your existing roles both for development and for administration.

In this scenario, there is no need to migrate your old developer roles as they are specific to XSC. The new developer roles will be created and managed with the XSA admin tools. The rest of the roles used to access SAP HANA (e.g. admin) can be migrated.

To migrate the roles the XS Advance Migration Assistance is available to convert the XSC roles into XSA roles. The migration assistant tool use as input a delivery unit (DU) in the XSC system where the roles are included, and it creates an XSA project containing the roles and all the necessary files – refer to SAP HANA XS Advanced Migration Guide.

For the migration, it is recommended to setup a dedicated XSA space for roles to be migrated. The idea is to do the initial migration of the roles in a single HDI container. After the migration segregate the roles and privileges into different HDI containers, spaces or organizations to fit the requirements.

The HDI container to be used for the deployment of the migrated roles should be equipped with similar privileges as the _SYS_REPO user to be able to deploy the roles.

Based on the previous recommendations, a possible migration process could be the following:

1. Use the migration assistant tool to create the XSA roles project.
2. Import the XSA roles project in SAP Web IDE.
3. Equip the HDI container with similar privileges as the _SYS_REPO user.
4. Review/fine-tune the roles and deploy on the _SYS_REPO like HDI container.