How To... Define Standard Roles for Administrators and Developers in SAP HANA Database

Applicable Releases:

SAP HANA 1.0 SPS07 (Revision 70 or higher)

Version 1.0
March 2014

For additional information contact:
richard.bremer@sap.com
# Document History

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

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

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>🌐</td>
<td>Note or Important</td>
</tr>
<tr>
<td>🟢</td>
<td>Example</td>
</tr>
<tr>
<td>🍃</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
Table of Contents

1. Scenario Introduction ................................................................................................................. 1
   1.1 Note ..................................................................................................................................... 1
   1.2 Structure of This Guide .......................................................................................................... 1
   1.3 Purpose of the Roles .............................................................................................................. 1
   1.4 Guiding Principles in Designing the Roles ............................................................................ 2

2. Prerequisites .............................................................................................................................. 3
   2.1 Packages in the repository .................................................................................................... 3
   2.2 System Configuration ........................................................................................................... 3

3. Preparing a Secure Development System .................................................................................. 5
   3.1 Proposed repository layout .................................................................................................. 6
      3.1.1 Customer/Vendor Packages ........................................................................................ 6
      3.1.2 Package system-local ................................................................................................... 7
      3.1.3 Security considerations for the system-local package .................................................. 8
      3.1.4 Real-life example .......................................................................................................... 8

4. High-Level Overview of Template roles ................................................................................... 9
   4.1 Typical Role Selection .......................................................................................................... 9

5. Proposed standard roles for SAP HANA systems ..................................................................... 11
   5.1 Roles for Role Creators ....................................................................................................... 12
      5.1.1 role_builder_native.hdbrole ......................................................................................... 12
      5.1.2 role_editor_imported.hdbrole ..................................................................................... 13
      5.1.3 role_builder.hdbrole .................................................................................................... 14
      5.1.4 security_developer_protected.hdbrole ........................................................................ 16
   5.2 Roles for Security Admins ..................................................................................................... 17
      5.2.1 Prerequisite: wrapper procedure for granting of roles ..................................................... 17
      5.2.2 User Admin role ........................................................................................................... 22
      5.2.3 User Admin role (unrestricted) ..................................................................................... 24
      5.2.4 Role Security_Admin_Basic ......................................................................................... 25
      5.2.5 Role Security_Admin_Troubleshooting ...................................................................... 26
      5.2.6 Role Security_Admin_Audit ........................................................................................ 28
      5.2.7 Role Security_Admin .................................................................................................... 28
      5.2.8 Role Security_Admin_Extended ................................................................................... 29
      5.2.9 Role Security_Admin_Disk_Encryption ........................................................................ 31
      5.2.10 Role Audit_Operator .................................................................................................. 32
   5.3 Roles for Database Administrators ......................................................................................... 32
      5.3.1 Role basic_admin ........................................................................................................... 33
      5.3.2 Role persistence_admin ................................................................................................. 34
      5.3.3 Role backup_admin ....................................................................................................... 35
      5.3.4 Role system_admin_generic .......................................................................................... 36
      5.3.5 Role system_admin_preinstalled ................................................................................... 37
      5.3.6 Role backup_operator ................................................................................................... 38
      5.3.7 Role database_monitoring ............................................................................................. 39
5.3.8 Role data_admin ............................................................................................................................ 39

5.4 Roles for Repository Administrators .................................................................................................. 41
  5.4.1 Role repo_manager .......................................................................................................................... 41
  5.4.2 Role repo_exporter .......................................................................................................................... 42
  5.4.3 Role repo_importer .......................................................................................................................... 44
  5.4.4 Role content_transport_manager ................................................................................................. 45
  5.4.5 Role content_transport_executor ................................................................................................. 47
  5.4.6 Role content_transport_source ................................................................................................... 47

5.5 Roles for Developers .......................................................................................................................... 48
  5.5.1 Role xs_app_dev_generic ............................................................................................................... 49
  5.5.2 Role xs_app_dev_public ............................................................................................................... 50

5.6 Roles for Data Modelers ..................................................................................................................... 51
  5.6.1 Role data_modeling_generic ....................................................................................................... 52
  5.6.2 Role data_modeling_public .......................................................................................................... 54
  5.6.3 Role model_preview_unrestricted .............................................................................................. 56
  5.6.4 Role data_modeling ...................................................................................................................... 58
  5.6.5 Role build_analytic_privilege_public ......................................................................................... 59

5.7 Roles for security developers ............................................................................................................. 60
  5.7.1 Role build_analytic_privilege_generic ....................................................................................... 60

5.8 Roles for End-Users ............................................................................................................................ 62
  5.8.1 Role reporting_generic ................................................................................................................. 62
  5.8.2 Other end-user privilege requirements ......................................................................................... 64

5.9 Roles for support .................................................................................................................................. 65
  5.9.1 Role basis_support_read ............................................................................................................... 65
  5.9.2 Role basis_support_trace .............................................................................................................. 66
  5.9.3 Role app_support_read .................................................................................................................. 67
  5.9.4 Role app_support_trace ................................................................................................................. 68
  5.9.5 Role app_support_read_data ........................................................................................................ 69
  5.9.6 Debugging ..................................................................................................................................... 71
  5.9.7 Further support roles .................................................................................................................... 71

5.10 Role / System-tier matrix .................................................................................................................. 71

6. Appendix ............................................................................................................................................. 73
1. Scenario Introduction

People keep asking us for standard role templates for administrative tasks and development tasks in SAP HANA database. With this document, we deliver and describe role templates for roles in the areas of

- Database administration
- Security administration
- Repository administration
- Data modeling and SAP HANA XS development
- Support tasks (basis and administration support)

1.1 Note

This document is not part of SAP’s product documentation. For an official reference, please use the SAP HANA Security Guide and the SAP HANA Developer Guide which are available at https://help.sap.com/hana_platform.

1.2 Structure of This Guide

The main intention of this How-To Guide is to provide documentation and source code of the proposed role templates. It therefore does not contain the step-by-step instructions you will find in most How-To Guides. The guide is structured in five main sections as follows:

- Section 1
  General information on the guide and its contents
- Section 2
  Preparatory configuration steps for SAP HANA systems if you want to build exactly the roles that we propose
- Section 3
  Considerations for setting up the development system (the SAP HANA repository) in a secure way. This is important, because the way of designing roles that we propose is based on repository development.
- Section 4
  In this guide, we introduce more than 30 role proposals. Section 4 gives a high-level overview of the template roles, helping you choose those roles that you will need in your system.
- Section 5
  Documentation and source code of all roles that we propose.
- Section 6 / Appendix
  Contains a delivery-unit export of all template roles described in this guide.

We must stress that for most roles you will not be able to use them without modification in your system, because many roles will need customer-specific privileges on database objects and repository packages.

1.3 Purpose of the Roles

The roles described in this document are considered templates. That is, we do not expect that customers can implement these roles without modifications.
The roles for database administrators are mostly complete, because the related tasks are often generic and do not require access to specific database objects in customer-specific database schemas.

Any roles related to development and repository management cannot be generic, since these roles will need to grant access to specific portions of the repository and the database catalog, which are specific to each project or customer.

### 1.4 Guiding Principles in Designing the Roles

When designing the roles described in this document, we followed the following guiding principles:

- Assume very strong security requirements
  - Very granular role structure
  - Strong control over granting of roles (e.g. only allow granting of end-user roles that have been designed by the security team – roles created by third parties are not considered end-user roles, they must be included in roles designed by the security team)
  - Strong control over developer rights, especially on security relevant packages such as the packages containing roles
- Administration of the database (basis administration) and of the development environment (repository) are strictly separated:
  - A database admin does not have access to the repository
  - A developer or administrator of the repository does not have administrative rights outside of the repository
- Security administration is strictly separated from overall database administration
  - Only security administrators can create or modify users or roles.
- We only work actively with repository roles:
  - Only repository roles may be newly created
  - Only repository roles may be granted to database users
  - The “ROLE ADMIN” privilege is not granted to any role or user
How To... Define Standard Roles for Administrators in SAP HANA Database

2. Prerequisites

Required/recommended expertise or prior knowledge

- Knowledge of SAP HANA Software on the release level of Support Package Stack (SPS) 7
- In particular, we assume that readers are familiar with the role concepts of SAP HANA database:
  - General understanding of “catalog roles” (roles that can be created using the “create role” SQL statement) and “repository roles” (roles that can be created as design-time object in the SAP HANA repository, and that have to be activated before they can be assigned to users)
  - Know how to create, modify, grant and revoke repository roles
- We also assume that readers have a general understanding of the different types of privileges in the database. Especially knowledge of the database’s system privileges is essential in order to understand the proposed role contents. The actions permitted by the system privileges are listed in the [SAP HANA SQL and System Views Reference](http://help.sap.com/hana/SAP_HANA_SQL_and_System_Views_Reference_en.pdf) – in the context of the GRANT statement.

Additional Documentation

- SAP Help Library – HANA Security Guide:
- SAP Help Library – HANA Developer’s Guide:
- SAP Help Library – HANA Administration Guide:
- SAP Help Library – HANA SQL Reference

2.1 Packages in the repository

Before importing and activating the roles provided with this document, the following packages must exist in the repository:

- **sap** – you cannot avoid having that one anyway
- **sap-templates** – this will be created during import, as our roles get placed there. But you can also create it before importing the roles. If you create it, make it a structural package.
- **system-local** – this package always exists in SAP HANA systems
- **system-local.private** – this package is also an integral part of HANA installations
- **system-local.public** – is not automatically created. Also create it as a structural package

In short: Create package **system-local.public** before importing the delivery unit with template roles

See section [3.1](#) for an explanation.

2.2 System Configuration

Change SQLScript mode to UNSECURE
- indexer.ini – section [repository] – set sqlscript_mode = unsecure

This setting is needed in order to activate the stored procedures that we include in the export. After the procedures are successfully activated, you can change the parameter back to its previous value.

Setting sqlscript_mode to "unsecure" enables the creation of read-write procedures and of procedures in definer mode in the database.

The parameter change takes effect immediately, a restart of the database processes is not required.
Preventing a Secure Development System

When setting up the database system – i.e. when creating the first user accounts for database administrators, developers, and end users; preparing the system for development projects, etc. – there are many things one can do wrong, and a few things one should better do right.

The following rules will serve you well in the area of role management. For this guide, and especially for the first item in the following item list, we target a customer situation in which security requirements are high, and in which no identity management solution exists that might handle the creation and assigning of roles in SAP HANA appropriately.

- First of all, we hereby decree that any role created in the SAP HANA System in your project shall be a repository role. And it naturally follows that the system privilege ROLE ADMIN is not to be granted to anyone in the system.
  
  Note: this first rule is not intended for customers that have an identity management solution in place which can handle the granting and revoking of SAP HANA roles. It is meant for customers who need to manage database roles purely by means provided natively with the SAP HANA system.

- Secondly, we must make sure that only authorized users create roles that are granted to end-users (i.e. we must prevent arbitrary developers from creating roles that may be directly granted to end-users).
  
  o Technically, this is not trivial because of the following three reasons:
    A) minimal development privileges are sufficient to create roles;
    B) there is no system privilege controlling the ability to activate repository roles;
    Roles can be created in any package of the repository; and
    C) the GRANT_ACTIVATED_ROLE-procedure does not support a “name space filter”.
  
  o Hence we will need to design a workaround. The easiest such solution has two properties as outlined below. Unfortunately, this suggested workaround will make it impossible to grant roles through the user editor of SAP HANA studio.
    
    ▪ Create dedicated packages for role development and control that only users that are supposed to create roles have development privileges on the role packages. We suggest a repository layout.
    
    ▪ To enable name-space filtering during the role assignment, create a wrapper procedure around the grant_activated_role procedure; you might implement a filter on the grantee-name, too, so that it is not possible to grant roles to oneself. We introduce such procedures in 5.2.1.

- Thirdly, we must make sure that users that can edit roles cannot edit their own roles.
  
  o Which means that there must be at least two packages with roles: one package with roles for role editors (and the role editors do not have developer rights in that package); and one package in which role editors can create new roles.
  
  o The role editors themselves must take care not to grant any development privileges on role packages to the roles they manage – because otherwise arbitrary developers might end up with developer rights on a role package.
  
  o In particular, this means that no developer may have access to the root package of the repository.
3.1 Proposed repository layout

It is very easy to create a messy package structure in the repository, especially in development systems. And this is partly because of a lack of knowledge of SAP’s proposed structuring of the repository content. In general, we point you at the SAP HANA Developer Guide which is available at https://help.sap.com/hana_platform. The following instructions are based on that document.

In our world, the repository only consists of three main packages which are created directly in the repository root:

- **Package sap** exists in any case – this is for official SAP content
- **Package <vendor>** : it is SAP’s suggestion that software producers (customers, vendors) create their own package in the root of the repository. If in a course of a customer project, the project team creates repository content such as virtual data models, SAP HANA XS Applications or similar, there should be a “<vendor>” package, where <vendor> would typically be the customer name. Any development that is supposed to be propagated to productive usage should happen within this package.
- **Package system-local** as a space for free-style testing, with three areas:
  - **Package system-local.public** as an open space for free-style testing – should only be used in non-production systems. All developers get full access to this space.
  - **Package system-local.private**: similar to public, but developers or teams get access only to a dedicated area underneath private
  - **Package system-local.generated**: package for generated content, that is, content not created by manual user interaction.

In the following sections, we will provide more details regarding these repository areas.

3.1.1 Customer/Vendor Packages

For developments that shall be promoted to production, SAP suggests that every “vendor” create their own package in the root of the repository tree.

Inside their <vendor> package, developers will create their own appropriate package structure. There is presently no generic guideline for this structuring.

For customer developments (i.e. not vendors producing content for others to implement, but rather in-house development at customers), we propose separating some security objects from the other content of vendor specific packages. These security objects are primarily all roles that are intended to be granted to end-users.

The reason for such separation is simply to protect security relevant developments from interference by regular developers.

Hence our proposed structure for the <vendor> package in in-house developments at customers is:

- `<vendor>`: root package for in-house developments at customer
- `<vendor>.security`: package for objects that are critical in the context of security

Inside of the `<vendor>.security` package, we even propose a further structure into two different portions:

- `<vendor>.security.protected`: roles granted to security developers (and related content)
- `<vendor>.security.common`: objects created by security developers

If you choose this package layout, it is mandatory that content developers do not get edit or activate privileges on the `<vendor>` package itself, but only on sub-packages. Otherwise, content
developers would implicitly get access to the <vendor>.security package, thus invalidating the entire layout in terms of security.

3.1.1 Special security considerations for application roles

While we are proposing a central package for role development at any customer site, we expect that naturally an application developer will create roles to interact with their application. We suggest taking the following approach:

- **Role creation:**
  create such roles locally in the application package. That’s the only logical place

- **Granting of roles to end-users:**
  Roles granted to end-users should be located in one or very few central packages. We suggest you do not grant the application roles directly to end-users. Instead, either create dedicated wrapper roles in <vendor>.security.common.roles – or include the application-local roles into your regular end-user roles. It is now the responsibility of your central role developers to verify the consistency and risk-level of the application-local roles that they include in end-user roles.

3.1.2 Special security considerations for analytic privileges

In our eyes, a developer of a data model should not have permission to create analytic privileges. These two acts should generally be separated.

In HANA, this only works if analytic privileges are not created in the same package as data models. Therefore, we suggest you create a dedicated package for analytic privileges. You have two natural choices:

- `<vendor>.<some_package>.privileges` – we don’t like that, because analytic privileges are security objects and thus belong into

- `<vendor>.security.common.privileges`
  Create analytic privileges and all related objects there. We suggest only using dynamic analytic privileges, which come with lookup-procedures and lookup-tables. All these objects can and should be created as repository-objects to make the privileges fully transportable.

3.1.2 Package system-local

Developers want to and need to play and test with the development platform they are using. We suggest a simple setup for such a testing ground:

A standard HANA installation initially contains a package named system-local. This package is foreseen by SAP for free-style testing, i.e. for development that is not supposed to be promoted to productive usage.

The system-local package and its content are non-transportable in the way that they cannot be attached to delivery units. Since all transport mechanisms are based on delivery units, transporting these free-style development objects is thus prohibited.

The system-local package comes with two pre-delivered structural sub-packages:

- **system-local.private**
  A more controlled testing area. Initially, no developer has access here. Individual developers or teams may request and then be provided packages here to which only they have access. Work done here will never be promoted into a production system, it’s purely intended for trial&error, and intentionally kept out of package <vendor>

It is important to keep system-local.private a structural package and create sub-packages for each user or team that requires a space for private testing.
How To... Define Standard Roles for Administrators in SAP HANA Database

- **system-local.generated**
  In this package, there may be generated content, i.e. content not created through manual interaction.

In addition to these two pre-delivered packages we propose creating a further structural sub-package named **system-local.public**.

- **system-local.public**
  Testing ground available to all developers. Everybody has full access, which means that everybody can see, edit, activate or delete any object in any package. It’s supposed to be used for chaotic and fruitful testing.
  In this area, people can (and should) create their own packages, and then build structures within these packages as they please.

### 3.1.3 Security considerations for the system-local package

If the private package in **system-local** shall really be used for private development – i.e. access to sub-packages shall be restricted to selected individuals – developers must not receive privileges on the system-local package itself.

This is also the reason for us to propose the creation of **system-local.public**. Developers shall have a space in which they can freely create new packages (and content of these packages), but we cannot allow them to create new packages directly in the system-local package, because then they would also have privilege on **system-local.private**.

### 3.1.4 Real-life example

Let us assume an imaginary customer NASA who produces down-to-earth consumer goods and children’s toys. We like to think that they grow potatoes and produce hammers and nails, and also beautiful vintage toys such as blue marbles and remote-controlled cars.

They will have application data in schema SAPNAP (Nasa Applicationdata Production) and they will create their own repository objects in packages underneath package **nasa**.

After they read our guide, NASA decided that the following package structure is very useful:

- **nasa.security** – any security-specific objects that may be granted to end-users.
- **nasa.security.common.roles** – The only place with NASA-developed end-user roles
- **nasa.security.common.privileges** – The only place with NASA-developed analytic privileges
- **nasa.security.protected.roles** – This is the place for critical roles (role editor etc.)
- **nasa.<other_packages>** – e.g. application, virtual data models, whatever else NASA builds...

At the same time, NASA have imported the roles accompanying this document, which ended up in

- **sap-templates.security**
- **sap-templates.security.common.roles**
- **sap-templates.security.protected.roles**
- **sap-templates.security.protected.procedures**
4. High-Level Overview of Template roles

We have designed the template roles rather detailed on purpose. The high granularity supports a highly specialized IT team setup and even if the roles may not fit perfectly the requirements of a given IT team, it will be easy to create appropriate roles for most circumstances.

At the same time, most IT teams will not require the full granularity that we introduce. Acknowledging this, we also introduce collection roles which we expect to be a good match in most cases.

In this section, we first present a short list of this best-fit selection of roles, and in the second part an overview of all roles and interdependencies between them.

4.1 Typical Role Selection

We expect that typically, you will need roles for the following user groups:

- **Security Developers**
  - who create roles and related objects
    - Role role_builder (5.1.3)
  - who may edit “protected” (security-critical roles)
    - Role security_developer_protected (5.1.4)
    - If you do not need the separation into “common” and “protected” security objects, your role developers may have both roles (role_builder and security_developer_protected)
  - who create Analytic Privileges (in the data mart use-case)
    - Role “build_analytic_privilege_generic” (5.7.1)
    - This role needs to be extended with project-specific details (package privileges and potentially data access privileges for creating dynamic Analytic Privileges)

- **Database administrators**
  - who manage the database system
    - Role system_admin_generic (5.3.4)
    - and at the same time
    - Role system_admin_preinstalled (5.3.5)
  - who control security parameters of the system (e.g. password policy)
    - Role security_admin (5.2.7)
  - who change audit settings of the database
    - Role audit_admin_audit (5.2.6)
  - who manage the audit trail “database table” (if used)
    - Role audit_operator (5.2.10)

- **User administrators**
  - who can create, drop, lock/unlock and modify users, including role granting (but who can only grant roles from the role packages we propose here)
    - Role user_admin (5.2.2)
  - as above, but unrestricted granting of all repository roles in the system (less secure, more convenient than user_admin)
    - Role user_admin_unrestricte (5.2.3)

- **Development/content administrators**
  - who can generally manage the repository itself
    - Role repo_manager (5.4.1)
  - who can set up transports of repository content
    - Role content_transport_manager (5.4.4)
  - who can execute transports (but not set up new transport routes)
Role content_transport_executor (5.4.5)
- who acts as technical user for providing the transport objects in the source system of the transport
Role content_transport_source (5.4.6)

- Data Modelers
  - who have all generically required privileges to create data models, plus the package privileges required to create models on the system-local.public package.
  - For development projects, the role needs to be extended with further package privileges on the project development packages.
  - Role data_modeling (5.6.4)
  - who build analytic privileges
    - As for data modelers, you will have to extend this role by package privileges on project-specific packages (repo.read, edit and activate on packages containing analytic privileges, repo.read on packages containing data models)
  - Role build_analytic_privilege_public (5.6.5)

- Developers (generic repository development)
  - who create e.g. XS applications
    - This role simply lists the minimal requirements for development. Further requirements may be data access to database tables, debugging privileges, etc. – which we cannot state generically. We recommend referring to the SAP HANA developer guide for more details
  - Role xs_app_dev_public (5.5.2)

- End-Users
  - who consume data models (reporting users)
  - Role reporting_generic (5.8.1)

- Support users
  - who can support database basis topics
    - Role basis_support_trace (5.9.2)
  - who can support application development topics, including data modeling
    - Role application_support_trace (5.9.4)
5. Proposed standard roles for SAP HANA systems

We have created a set of generic roles that will be useful in SAP HANA systems. There are roles for multiple groups of users:

- **ROLES FOR ROLE CREATORS**
- **ROLES FOR SECURITY ADMINS**
- **ROLES FOR DATABASE ADMINISTRATORS**
- **ROLES FOR REPOSITORY ADMINISTRATORS**
- **ROLES FOR DEVELOPERS**
- **ROLES FOR DATA MODELERS**
- **ROLES FOR SECURITY DEVELOPERS**
- **ROLES FOR END-USERS**
- **ROLES FOR SUPPORT**

**Important Note**

Generic roles means that the role definitions do not contain any reference to application data or packages. Only references to system-provided schemas such as the schemas _SYS_STATISTICS, _SYS_SECURITY, _SYS_BI or _SYS_BIC are allowed. And on the repository side, there will only be references to repository objects in the sap package (e.g. sap.hana.xs.lm.roles) or references to the roles listed in this guide.

For this guide, we assume that our proposed roles are located in package /sap-templates/security with the following layout:

- **sap-templates.security**
  - **sap-templates.security.protected.roles** – protected space for sensitive roles
  - **sap-templates.security.protected.procedures** – protected space for helper procedures
- **sap-templates.security.common.roles** – space in which role creators can work
- **sap-templates.security.common.roles.admin** – generic roles for admin and monitoring
- **sap-templates.security.common.roles.security** – generic roles for security admins
- **sap-templates.security.common.roles.development** – generic roles for developers (XS, vdm, ...)
- **sap-templates.security.common.roles.endusers** – generic roles for end users
- **sap-templates.security.common.roles.support.basis** – generic roles for DB basis support
- **sap-templates.security.common.roles.support.app** – generic roles for application support

In addition to these generic roles, which are largely independent of the actual usage of SAP HANA, there will always be project-specific roles. These roles will contain references to data objects (schemas with application data, repository packages with customer-specific content such as data models or applications, ...).

We suggest placing such customer-specific roles into sub-packages of your own <vendor> package.
5.1 Roles for Role Creators

An SAP-delivered role for role creators does not make too much sense, because a) we do not know our customer’s repository layout; and b) customers should not edit SAP-delivered roles.

Still, of course a situation may arise in which customers want to either add roles among those delivered with this guide, or to modify some of these roles. In those cases, we do, however, encourage you to create copies of our roles inside your own development package. Therefore this role is mostly a template for the customer-specific role-creator-role which in NASA’s case would be placed into package /nasa/security/protected/roles and enable role editing in package /nasa/security/common/roles.

5.1.1 role_builder_native.hdbrole

The role_builder_native role allows general development of native objects in package /sap-templates/security/common/roles. That is, a holder of this role can create new roles or packages in that package.

5.1.1.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>REPO.READ on package sap-templates.security.common.roles</td>
<td>Read content of roles (and other development objects)</td>
</tr>
<tr>
<td>REPO.EDIT_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Create or modify locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Activate locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_NATIVE_PACKAGES on package sap-templates.security.common.roles</td>
<td>Create, modify or drop locally created (not imported) packages</td>
</tr>
</tbody>
</table>

5.1.1.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Role for a person that is allowed to
// create new roles in a dedicated package
// <role_package> of the SAP HANA Repository,
// and to modify such newly created roles,
// but who is not allowed to modify imported
// roles in that package.
//
// Note that anyone that is assigned to this role
// must not have any role from <role_package>.
//
// Here: <role_package> = sap-templates.security.common.roles
//
// actions enabled by this role:
```

```plaintext
```
5.1.2 role_editor_imported.hdbrole

The role_editor_imported role allows general development on imported objects in package /sap-templates/security/common/roles. That is, a holder of this role is allowed to modify the SAP-delivered objects in that package.

Basically, if you really wanted to, you could use this role to modify the roles we deliver via this document. We rather recommend creating copies of these roles in your own package.

5.1.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>REPO.READ on package sap-templates.security.common.roles</td>
<td>Read content of roles (and other development objects)</td>
</tr>
<tr>
<td>REPO.EDIT_IMPORTED_OBJECTS on package sap-templates.security.common.roles</td>
<td>Create or modify locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_IMPORTED_OBJECTS on package sap-templates.security.common.roles</td>
<td>Activate locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_IMPORTED_PACKAGES on package sap-templates.security.common.roles</td>
<td>Create, modify or drop locally created (not imported) packages</td>
</tr>
</tbody>
</table>
How To... Define Standard Roles for Administrators in SAP HANA Database

5.1.2.2 hdbrole file

The hdbrole file looks as follows:

```hdbrole
// Role for a person that is allowed to
// modify imported roles in a dedicated package
// <role_package> of the SAP HANA Repository,
// but who is not allowed to create new roles
// in that package.
//
// Note that anyone that is assigned to this role
// must not have any role from <role_package>.
//
// Here: <role_package> = sap-templates.security.common.roles
//
// actions enabled by this role:
// - Role "role_editor_imported"
//  * access to the repository stored procedure
//  * work in repo-package sap-templates.security.common.roles:
//    + read repo content
//    + edit repo content (imported objects)
//    + activate repo content (imported objects)
//    + maintain imported packages
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::role_editor_imported {
    -- general access to the repository stored procedure:
    catalog sql object "PUBLIC"."REPOSITORY_REST": EXECUTE;
    -- read, edit and activate objects in
    -- the package for general roles
    package sap-templates.security.common.roles: REPO.READ;
    package sap-templates.security.common.roles: REPO.EDIT_IMPORTED_OBJECTS;
    package sap-templates.security.common.roles: REPO.ACTIVATE_IMPORTED_OBJECTS;
    -- and also permission to manage packages, e.g. create new sub-packages
    -- for defining a package-based naming convention
    package sap-templates.security.common.roles: REPO.MAINTAIN_IMPORTED_PACKAGES;
}
```

5.1.3 role_builder.hdbrole

The role_builder role allows general development in package
/sap-templates/security/roles, both on newly created as well as imported objects. It is thus
simply a combination of roles role_builder_native and role_editor_imported. It is the role you need if
you want to edit our template roles and create new roles within the same package structure.

5.1.3.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>REPO.READ on package sap-templates.security.common.roles</td>
<td>Read content of roles (and other development objects)</td>
</tr>
</tbody>
</table>
How To... Define Standard Roles for Administrators in SAP HANA Database

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPO.EDIT_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Create or modify locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Activate locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_NATIVE_PACKAGES on package sap-templates.security.common.roles</td>
<td>Create, modify or drop locally created (not imported) packages</td>
</tr>
<tr>
<td>REPO.EDIT_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Create or modify locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_NATIVE_OBJECTS on package sap-templates.security.common.roles</td>
<td>Activate locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_NATIVE_PACKAGES on package sap-templates.security.common.roles</td>
<td>Create, modify or drop locally created (not imported) packages</td>
</tr>
</tbody>
</table>

5.1.3.2  hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Role for a person that is allowed to
// create new roles in a dedicated package
// <role_package> of the SAP HANA Repository,
// and to modify such newly created roles or
// existing (imported) roles in that package.
//
// Note that anyone that is assigned to this role
// must not have any role from <role_package>.
//
// Here: <role_package> = sap-templates.security.common.roles
//
// actions enabled by this role:
// - Role "role_builder_native"
//   * access to the repository stored procedure
//   * work in repo-package sap-templates.security.common.roles:
//     + read repo content
//     + edit repo content (native objects)
//     + activate repo content (native objects)
//     + maintain native packages
// - Role "role_editor_imported"
//   * access to the repository stored procedure
//   * work in repo-package sap-templates.security.common.roles:
//     + read repo content
//     + edit repo content (imported objects)
//     + activate repo content (imported objects)
//     + maintain imported packages
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::role_builder
extends role sap-templates.security.protected.roles::role_builder_native,
  sap-templates.security.protected.roles::role_editor_imported
```

15
5.1.4 security_developer_protected.hdbrole

By separating the role packages into the “protected” and “common” area, we introduced protection from role editors modifying their own roles. However, you might need to be able to change the content of the “protected” package. This role allows changing and activating both native and imported objects, as well as modifying the package structure underneath “protected”. In a strictly controlled environment, you should only make use of this role during system setup and in “emergency situations”.

5.1.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>REPO.READ on package sap-templates.security.protected</td>
<td>Read content of roles (and other development objects)</td>
</tr>
<tr>
<td>REPO.EDIT_NATIVE_OBJECTS on package sap-templates.security.protected</td>
<td>Create or modify locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_NATIVE_OBJECTS on package sap-templates.security.protected</td>
<td>Activate locally created (not imported) objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_NATIVE_PACKAGES on package sap-templates.security.protected</td>
<td>Create, modify or drop locally created (not imported) packages</td>
</tr>
<tr>
<td>REPO.EDIT_IMPORTED_OBJECTS on package sap-templates.security.common.roles</td>
<td>Create or modify imported objects</td>
</tr>
<tr>
<td>REPO.ACTIVATE_IMPORTED_OBJECTS on package sap-templates.security.protected</td>
<td>Activate imported objects</td>
</tr>
<tr>
<td>REPO.MAINTAIN_IMPORTED_PACKAGES on package sap-templates.security.protected</td>
<td>Create, modify or drop imported packages</td>
</tr>
</tbody>
</table>

5.1.4.2 hdbrole file

The .hdbrole file looks as follows:

```java
// // Role for a person that is allowed to
// create new security-related objects
// in the "protected" space (<protected_package>)
// for security-critical roles and other objects
// The regular "role_builder" roles do not have
// these privileges, to prevent role editors from
// modifying their own roles.
// In our template-package, this role allows creating,
// editing and activating native and imported objects
// within the sap-templates.security.protected space
// Here: <protected_package> = sap-templates.security.protected
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.2 Roles for Security Admins

We provide roles mainly for two types of security admins:

- People who create and manage database users (role user_admin)
- People who manage security-relevant settings of the database

5.2.1 Prerequisite: wrapper procedure for granting of roles

Before diving into the first role definitions, we have to build a few procedures that help us create safer user management roles. Especially, we define procedures that restrict granting of roles to roles from certain packages.

Note

By using our proposed stored procedure wrappers (as opposed to directly using the “GRANT_ACTIVATED_ROLE” procedure), you disable granting of repository roles via SAP HANA Studio or the Security UI of the SAP HANA XS web IDE.
5.2.1.1 Procedure source code - roles in common package

This procedure can be used to grant any role that's in a sub-package of sap-templates.security.common.roles. It accepts two parameters:

- **role_name**
  This is the name of the role and the sub-package of sap-templates.security.common.roles in which the role resides.
  For role sap-templates.security.common.roles.admin::monitoring, the value of role_name must be admin::monitoring

- **grantee**
  Name of the database user to which you want to grant the role.

The procedure is simply a wrapper around the GRANT_ACTIVATED_ROLE procedure delivered with SAP HANA. We check the following conditions and throw errors if they are violated:

- Grantee must exist error code 11001
- Grantee must be different from grantor error code 11002
- Role must exist error code 11003

We include the EXECUTE privilege for this role in our suggested role sap-templates.security.protected.roles::user_admin

```sql
PROCEDURE "SYS_BIC"."sap-templates.security.protected.procedures::grant_sap_common_role" ( IN role_name VARCHAR(128), grantee VARCHAR(128) )
LANGUAGE SQLSCRIPT
SQL SECURITY DEFINER
AS
-- SQL statement we're going to execute
v_statement VARCHAR(256);
-- base package for all roles
v_basePackage VARCHAR(128) := 'sap-templates.security.common.roles';
-- we assemble the role name from base package and in-param ROLE_NAME
v_fqRoleName VARCHAR(256);
found INT := 0;
BEGIN
-- prepare error handling in case of invalid arguments
DECLARE USERNOTEXIST CONDITION FOR SQL_ERROR_CODE 11001;
DECLARE GRANTSELF CONDITION FOR SQL_ERROR_CODE 11002;
DECLARE ROLENOTEXIST CONDITION FOR SQL_ERROR_CODE 11003;
DECLARE EXIT HANDLER FOR USERNOTEXIST RESIGNAL;
DECLARE EXIT HANDLER FOR GRANTSELF RESIGNAL;
DECLARE EXIT HANDLER FOR ROLENOTEXIST RESIGNAL;

-- create the fully qualified role name
-- we assume that all roles will always be
-- v_basePackage.<package>::<role>
-- i.e. no roles are directly in v_basePackage
v_fqRoleName := :v_basePackage || '.' || role_name;

-- check input parameter role:
-- does the role exist?
SELECT COUNT(*) INTO found FROM "ROLES"
WHERE "ROLE_NAME" = :v_fqRoleName;
IF :found = 0 THEN
  SIGNAL ROLENOTEXIST SET MESSAGE_TEXT =
  'Role does not exist: ' || :v_fqRoleName;
END IF;
```
How To... Define Standard Roles for Administrators in SAP HANA Database

-- check input parameter user:
-- does grantee exist?
SELECT COUNT(*) INTO found FROM "USERS"
    WHERE "USER_NAME" = :grantee;
IF :found = 0 THEN
    SIGNAL USERNOTEXIST SET MESSAGE_TEXT =
        'Grantee does not exist: ' || :grantee;
END IF;
-- self grant?
IF :grantee = SESSION_USER THEN
    SIGNAL GRANTSELF SET MESSAGE_TEXT =
        'Self-grant not allowed';
END IF;
-- assemble grant statement: we have to call the
-- GRANT_ACTIVATED_ROLE procedure
v_statement :=
    'CALL GRANT_ACTIVATED_ROLE (''' ||
    :v_fqRoleName || ''', ''' || :grantee || '''');
-- and run the statement:
EXEC v_statement;
END;

Invocation of the procedure:
CALL _SYS_BIC."sap-templates.security.protected.procedures::grant_sap_common_role"
    ('<sub_package>::<role_name>', '<grantee>')
-- example:
CALL _SYS_BIC."sap-templates.security.protected.procedures::grant_sap_common_role"
    ('admin::database_monitoring', 'MONITORING_USER')
-- grant role sap-templates.security.common.roles.admin::database_monitoring to
-- user MONITORING_USER

5.2.1.2  Procedure source code – roles in protected package
This procedure can be used to grant any role that’s located directly in package sap-templates.security.protected.roles. It accepts two parameters:

- role_name
  This is the name of the role (without package)
  For role sap-templates.security.protected.roles::role_builder, the value of role_name must simply be role_builder

- grantee
  Name of the database user to which you want to grant the role.

The procedure is simply a wrapper around the GRANT_ACTIVATED_ROLE procedure delivered with SAP HANA. We check the following conditions and throw errors if they are violated:

- Grantee must exist  error code 11001
- Grantee must be different from grantor  error code 11002
- Role must exist  error code 11003

We include the EXECUTE privilege for this role in our suggested role sap-templates.security.protected.roles::user_admin
PROCEDURE "SYS_BIC"."sap-templates.security.protected.procedures::grant_sap_protected_role"
  ( IN role_name VARCHAR(128), grantee VARCHAR(128) )
LANGUAGE SQLSCRIPT
SQL SECURITY DEFINER
AS
  -- SQL statement we're going to execute
  v_statement VARCHAR(256);
  -- base package for all roles
  v_basePackage VARCHAR(128) := 'sap-templates.security.protected.roles';
  -- we assemble the role name from base package and in-param ROLE_NAME
  v fqRoleName VARCHAR(256);
  found INT := 0;
BEGIN
  -- prepare error handling in case of invalid arguments
  DECLARE USERNOTEXIST CONDITION FOR SQL_ERROR_CODE 11001;
  DECLARE GRANTSELF CONDITION FOR SQL_ERROR_CODE 11002;
  DECLARE ROLENOTEXIST CONDITION FOR SQL_ERROR_CODE 11003;
  DECLARE EXIT HANDLER FOR USERNOTEXIST RESIGNAL;
  DECLARE EXIT HANDLER FOR GRANTSELF RESIGNAL;
  DECLARE EXIT HANDLER FOR ROLENOTEXIST RESIGNAL;

  -- create the fully qualified role name
  -- we assume that all roles will always be
  -- v_basePackage::<role>
  -- i.e. all roles are directly in v_basePackage
  v fqRoleName := :v_basePackage || '::' || role_name;

  -- check input parameter role:
  -- does the role exist?
  SELECT COUNT(*) INTO found FROM "ROLES"
    WHERE "ROLE_NAME" = :v fqRoleName;
  IF :found = 0 THEN
    SIGNAL ROLENOTEXIST SET MESSAGE_TEXT =
      'Role does not exist: ' || :v fqRoleName;
  END IF;

  -- check input parameter user:
  -- does grantee exist?
  SELECT COUNT(*) INTO found FROM "USERS"
    WHERE "USER_NAME" = :grantee;
  IF :found = 0 THEN
    SIGNAL USERNOTEXIST SET MESSAGE_TEXT =
      'Grantee does not exist: ' || :grantee;
  END IF;

  -- self grant?
  IF :grantee = SESSION_USER THEN
    SIGNAL GRANTSELF SET MESSAGE_TEXT =
      'Self-grant not allowed';
  END IF;

  -- assemble grant statement: we have to call the
  -- GRANT_ACTIVATED_ROLE procedure
  v_statement :=
    'CALL GRANT_ACTIVATED_ROLE (''' ||
    :v fqRoleName || ''' ||
    :v fqRoleName || ''' ||
    :grantee || '''
)';
How To... Define Standard Roles for Administrators in SAP HANA Database

5.2.1.3 Procedure source code: grant SAP_INTERNAL_HANA_SUPPORT role

HANA comes with a very special and powerful role that should only be granted to SAP development support staff in exceptional circumstances.

This role has the following properties:

- It is a catalog role (we don’t like that)
- It can only be granted to one user at a time
- It cannot be granted to other roles (catalog or repository)

Because of the above properties, the role can only be granted by a DB user with the ROLE ADMIN privilege – and we don’t want to grant that one in our HANA system.

But _SYS_REPO has ROLE ADMIN, so we can create a stored procedure in definer mode to grant the role.

We include the EXECUTE privilege for this role in our suggested role sap-templates.security.protected.roles::user_admin

```sql
PROCEDURE "_SYS_BIC"."sap-templates.security.protected.procedures::grant_sap_internal_hana_support_role" (IN GRANTEE VARCHAR(128) )
LANGUAGE SQLSCRIPT
SQL SECURITY DEFINER
AS
v_statement VARCHAR(256);
found INT := 0;
BEGIN
-- prepare error handling in case of invalid grantee
DECLARE USERNOTEXIST CONDITION FOR SQL_ERROR_CODE 11001;
DECLARE EXIT HANDLER FOR USERNOTEXIST RESIGNAL;
-- check input parameter
SELECT COUNT(*) INTO found FROM "USERS"
WHERE "USER_NAME" = :grantee;
IF :found = 1 THEN
  v_statement :=
    'GRANT SAP_INTERNAL_HANA_SUPPORT TO ' || :GRANTEE;
  EXEC v_statement;
ELSE
```

Invocation of the procedure:

```sql
CALL _SYS_BIC."sap-templates.security.protected.procedures::grant_sap_protected_role"
('<role_name>', '<grantee>')
-- example:
CALL _SYS_BIC."sap-templates.security.protected.procedures::grant_sap_protected_role"
('role_builder', 'ROLE_CREATOR_USER')
-- grant role sap-templates.security.protected.roles::role_builder to
-- user ROLE_CREATOR_USER
```
5.2.1.4 Procedure source code: revoke SAP_INTERNAL_HANA_SUPPORT role

Same as above, but revoking of the role. We don’t need to check that the “grantee” already has the role, the statement will always be considered successful.

We include the EXECUTE privilege for this role in our suggested role sap-templates.security.protected.roles::user_admin

```
PROCEDURE "_SYS_BIC"."sap-templates.security.protected.procedures::revoke_sap_internal_hana_support_role"
    (IN GRANTEE VARCHAR(128) )
    LANGUAGE SQLSCRIPT
    SQL SECURITY DEFINER
    AS
    v_statement VARCHAR(256);
    found INT := 0;
    BEGIN
        -- prepare error handling in case of invalid grantee
        DECLARE USERNOTEXIST CONDITION FOR SQL_ERROR_CODE 11001;
        DECLARE EXIT HANDLER FOR USERNOTEXIST RESIGNAL;
        -- check input parameter
        SELECT COUNT(*) INTO found FROM "USERS"
        WHERE "USER_NAME" = :grantee;
        IF :found = 1 THEN
            v_statement :=
                'REVOKE SAP_INTERNAL_HANA_SUPPORT FROM ' || :GRANTEE;
            EXEC v_statement;
        ELSE
            SIGNAL USERNOTEXIST SET MESSAGE_TEXT =
                'INVALID GRANTEE PROVIDED' ;
        END IF;
    END;
```

5.2.2 User Admin role

Our user admin is allowed to create new users, and to grant roles to them. In our case, the granting of roles is restricted to repository roles; and even there, only to those roles that are in the package of SAP-suggested security objects sap-templates.security

We include here also the privilege to grant roles that might be considered sensitive, such as user admin, security admin and role admin. If not desired, simply create a copy of this role, but exclude the EXECUTE privilege on sap-templates.security.protected.procedures::grant_sap_protected_role

Note that it increases system security if only roles from a specifically controlled package tree are grantable. At the same time, however, you cannot make use of the standard procedures for granting and revoking anymore, and this also means that the user editor in SAP HANA studio cannot be used for granting or revoking roles if you choose this approach.
If you do not need this extra security, you may choose the role user_admin_unrestricted from the next section.

### 5.2.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege: CATALOG READ</td>
<td>Access to metadata of the database catalog – useful but not absolutely necessary.</td>
</tr>
<tr>
<td>System privilege: USER ADMIN</td>
<td>Ability to create, modify, drop, lock and unlock users</td>
</tr>
<tr>
<td>EXECUTE on procedure grant_sap_common_role in package sap-templates.security.protected.procedures</td>
<td>Allows granting of roles in the “common” package of SAP-suggested roles – almost all roles are here.</td>
</tr>
<tr>
<td>EXECUTE on procedure grant_sap_protected_role in package sap-templates.security.protected.procedures</td>
<td>Allows granting of a small number of roles which we placed in the protected package of SAP-suggested roles. These roles are of a sensitive nature, e.g. role_builder or security_admin</td>
</tr>
<tr>
<td>EXECUTE on procedure PUBLIC.REVOKE_ACTIVATED_ROLE</td>
<td>Allows revoking any repository role in the system</td>
</tr>
<tr>
<td>EXECUTE on procedure grant_sap_internal_hana_support_role in package sap-templates.security.protected.procedures</td>
<td>Allows granting of the very special SAP_INTERNAL_HANA_SUPPORT role to a user if this is really required.</td>
</tr>
<tr>
<td>EXECUTE on procedure revoke_sap_internal_hana_support_role in package sap-templates.security.protected.procedures</td>
<td>Allows revoking of the very special SAP_INTERNAL_HANA_SUPPORT role to a user if this is really required.</td>
</tr>
</tbody>
</table>

### 5.2.2.2 hdbrole file

The .hdbrole file looks as follows:

```bash
// Role for DB user who can
// - create new user accounts
// - modify user accounts
// - grant activated (repository) roles in packages of
//   sap-suggested roles (/sap-templates/security) to users
// - revoke any activated role from users
// - grant and revoke the SAP_INTERNAL_HANA_SUPPORT_ROLE
//
// Explicitly not included:
// - any direct granting of privileges
// - granting of catalog roles
// - any development privilege
// - unfiltered granting of all roles (procedure GRANT_ACTIVATED_ROLE)
//
// Author: Richard Bremer <richard-bremer@sap.com>
role sap-templates.security.protected.roles::user_admin {
  // for convenience: add catalog read
  system privilege: CATALOG READ;
  // user admin allows creation, dropping, modification of users
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.2.3 User Admin role (unrestricted)

The user admin role in the previous section is restricted to granting only those roles that we propose here. If you want to or need to grant arbitrary roles in the system, you may use the role user_admin_unrestricted which allows granting any repository role.

Note that our role templates are based on the assumption that you work only with repository roles and thus we do not include privileges to grant catalog roles. If you need catalog roles, you may include them in a wrapper repository role.

5.2.3.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Access to metadata of the database catalog – useful but not absolutely necessary.</td>
</tr>
<tr>
<td>System privilege: USER ADMIN</td>
<td>Ability to create, modify, drop, lock and unlock users</td>
</tr>
<tr>
<td>EXECUTE on procedure PUBLIC.GRANT_ACTIVATED_ROLE</td>
<td>Allows granting any repository role in the system</td>
</tr>
<tr>
<td>EXECUTE on procedure PUBLIC.REVOKE_ACTIVATED_ROLE</td>
<td>Allows revoking any repository role in the system</td>
</tr>
<tr>
<td>EXECUTE on procedure grant_sap_internal_hana_support_role in package sap-templates.security.protected.procedures</td>
<td>Allows granting of the very special SAP_INTERNAL_HANA_SUPPORT role to a user if this is really required.</td>
</tr>
<tr>
<td>EXECUTE on procedure revoke_sap_internal_hana_support_role in package sap-templates.security.protected.procedures</td>
<td>Allows revoking of the very special SAP_INTERNAL_HANA_SUPPORT role to a user if this is really required.</td>
</tr>
</tbody>
</table>

5.2.3.2 hdbrole file

The .hdbrole file looks as follows:

```sql
// including lock/unlock etc.

system privilege: USER ADMIN;
// allow granting of SAP-suggested standard roles (common space)
sql object sap-templates.security.protected.procedures::grant_sap_common_role: EXECUTE;
// allow granting of SAP-suggested standard roles (protected space):
sql object sap-templates.security.protected.procedures::grant_sap_protected_role: EXECUTE;
// allow revoking of repository roles:
catalog sql object "PUBLIC"."REVOKE_ACTIVATED_ROLE": EXECUTE;
// allow granting of SAP_INTERNAL_HANA_SUPPORT_ROLE
sql object sap-templates.security.protected.procedures::grant_sap_internal_hana_support_role: EXECUTE;
// allow revoking of SAP_INTERNAL_HANA_SUPPORT_ROLE
sql object sap-templates.security.protected.procedures::revoke_sap_internal_hana_support_role: EXECUTE;
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.2.4 Role Security_Admin_Basic

For a security admin, we identified a number of essential privileges, and a few optional ones. This role security_admin_basic collects the essential privileges. Below you find roles security_admin_troubleshooting and security_admin_audit; which enable further actions a security admin might need to perform.

We also implemented two collection roles, named security_admin and security_admin_extended, both of which include security_admin_basic.

Our security admin (basic) can modify security-related properties of the database, such as modifying the password policy, or managing the password blacklist.

5.2.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to the metadata of the catalog</td>
</tr>
<tr>
<td>System privilege INIFILE ADMIN</td>
<td>Here: required in order to modify the password blacklist; Undesirable but presently unavoidable side effect: can also modify any other system parameter.</td>
</tr>
</tbody>
</table>
How To... Define Standard Roles for Administrators in SAP HANA Database

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT, INSERT, UPDATE, DELETE on table _SYS_PASSWORD_BLACKLIST in schema _SYS_SECURITY</td>
<td>Read and modify the password blacklist</td>
</tr>
</tbody>
</table>

5.2.4.2 hdbrole file

The hdbrole file looks as follows:

```sql
// Role for persons who manages security properties of the database system.
//
// This role only contains privileges that the security admin will certainly require:
//
// Actions permitted by this role:
// - Role "security_admin_basic":
//   * Enter administration console in SAP HANA Studio
//     + This also gives read access to the metadata of the database catalog
//     + and read access to the DB configuration
//   * Modify the password policy
//     + side effect: this gives full write access to DB system configuration (inifiles)
//
// There are further security-admin-related privileges collected in roles security_admin_troubleshooting, security_admin_disk_encryption, and security_admin_audit, see also composite roles security_admin and security_admin_extended
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin_basic {
  -- enter into administration console, view system configuration and traces:
  system privilege: CATALOG READ;

  -- See and modify password blacklist:
  catalog sql object "SYS_SECURITY"."SYS_PASSWORD_BLACKLIST": SELECT, INSERT, UPDATE, DELETE;

  -- Modify password policy:
  -- side effect: full write access to DB system configuration (inifiles)
  system privilege: INIFILE ADMIN;
}
```

5.2.5 Role Security_Admin_Troubleshooting

This role assembles further privileges that can be useful for a security admin, but are not necessarily essential. Presently, the role empowers the user to start and stop traces – the security admin would be especially interested in authorization traces or traces related to the handling of analytic privilege (both part of the DB trace or user-specific trace), but this cannot be controlled on that granularity, so the security admin with this privilege will have full administrative rights on the database traces.
How To... Define Standard Roles for Administrators in SAP HANA Database

5.2.5.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to the metadata of the catalog</td>
</tr>
<tr>
<td>System privilege TRACE ADMIN</td>
<td>Here: empowers the user to run security-related traces, such as authorization trace. Unavoidable side-effect: the user can start/stop any trace in the system, not only security-related ones.</td>
</tr>
</tbody>
</table>

5.2.5.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Role for persons who manages security properties
// of the database system.
//
// This role only contains privileges that not all security admins may need and that have side-effects
//
// - Role "security_admin_troubleshooting":
//   * Enter administration console in SAP HANA Studio
//     + This also gives read access to the metadata of the database catalog
//   * run and analyze authorization traces
//     + this also allows changing any other trace setting
//
// There are further security-admin-related privileges collected in other roles, see role security_admin_extended and roles therein.
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin_troubleshooting
{
    -- enter into administration console, view system configuration
    -- and traces.
    system privilege: CATALOG READ;

    -- change trace settings, e.g. for authorization traces:
    -- side effect: can also run other traces, delete trace files, ...
    system privilege: TRACE ADMIN;

    -- for analyzing authorization issues, this user might
    -- also need read access to analytic privilege definitions
    -- in the repository (plus potentially procedures/lookup tables
    -- for dynamic APs); and read access to XS applications for
    -- understanding application privileges.
    -- These are, however, project-specific privileges.
}
```
5.2.6  Role Security_Admin_Audit

Our basic security admin has no privileges to manage the database’s audit setting. Instead, we propose a dedicated role for this purpose. We have included this role in the most complete collecting role security_admin_extended (5.2.8).

Note that we separate administration of the audit from reading (and managing) the content of the audit trail. See role audit_operator () for that purpose.

5.2.6.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to the metadata of the catalog</td>
</tr>
<tr>
<td>System privilege AUDIT ADMIN</td>
<td>Enable/disable auditing, change audit trail, add, modify or delete audit policies</td>
</tr>
</tbody>
</table>

5.2.6.2  hdbrole file

The .hdbrole file looks as follows:

```
// Role for persons who manage audit settings of the database.
//
// This role only contains privileges that are required to
// manage the audit settings of the database:
//
// Actions permitted by this role:
//  - Role "security_admin_audit":
//    * Read access to catalog metadata (e.g. table names)
//    + This also allows to enter the administration console
//      in SAP HANA Studio
//    + and read access to the DB configuration
//    * View audit settings
//    * Create, modify, delete audit policies
//
// There are further security-admin-related privileges collected in
// roles security_admin_basic and security_admin_optional.
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin_audit {
    -- view audit policies, create, modify, delete audit policies
    system privilege: AUDIT ADMIN;

    -- read catalog metadata, such as names of schemas and tables
    -- in the database (required for audit policies restricting
    -- on individual objects)
    -- side effect: read system configuration or
    system privilege: CATALOG READ;
}
```

5.2.7  Role Security_Admin

We assume that in most cases you will at least grant the basic and the troubleshooting privileges to your security admins. Hence we provide a role which combines both roles and might be standard role for security admins.
5.2.7.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role security_admin_basic</td>
<td>See 5.2.4</td>
</tr>
<tr>
<td>Role security_admin_troubleshooting</td>
<td>See 5.2.5</td>
</tr>
</tbody>
</table>

5.2.7.2 hdbrole file

The _hdbrole_ file looks as follows:

```plaintext
// Role for persons who manage security properties
// of the database system.

// We split the privileges into three portions, one for
// unquestionable privileges, one for optional privileges
// of security admins, and one for managing audit settings.

// This role here combines the first two portions:

// Actions permitted by this role:
// - Role "security_admin_basic":
//   * Enter administration console in SAP HANA Studio
//     + This also gives read access to the metadata of the database
catalog
//     + and read access to the DB configuration
//   * Modify the password policy
//     + side effect: this gives full write access to DB system
configuration (inifiles)

// - Role "security_admin_troubleshooting":
//   * Enter administration console in SAP HANA Studio
//     + This also gives read access to the metadata of the database
catalog
//     + and read access to the DB configuration
//   * run and analyze authorization traces
//     + this also allows changing any other trace setting

// further related privileges in security_admin_disk_encryption and
// security_admin_audit

// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin
  extends role sap-templates.security.protected.roles::security_admin_basic,
  sap-templates.security.protected.roles::security_admin_troubleshooting
{
  -- we add no individual privileges here.
}
```

5.2.8 Role Security_Admin_Extended

We assume that in most cases you will grant the basic and the optional privileges to your security admins. Hence we provide a role which combines both roles and might be standard role for security admins.
5.2.8.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role security_admin</td>
<td>See 5.2.7</td>
</tr>
<tr>
<td>Role security_admin_audit</td>
<td>See 5.2.6</td>
</tr>
</tbody>
</table>

5.2.8.2 hdbrole file

The `.hdbrole` file looks as follows:

```markdown
// Role for persons who manage security properties
// of the database system.

// We split the privileges into three portions, one for
// unquestionable privileges, and one for optional privileges,
// and one for audit-related privileges.

// The first two portions are added in role security_admin, which in turn
// combines roles security_admin_basic and security_admin_optional

// The third portion is contained in role security_admin_audit

// Actions permitted by this role:
// - Role "security_admin_basic":
//   * Enter administration console in SAP HANA Studio
//   + This also gives read access to the metadata of the database
catalog
//   + and read access to the DB configuration
//   + Modify the password policy
//   + side effect: this gives full write access to DB system
//     configuration (inifiles)

// - Role "security_admin_troubleshooting":
//   * Enter administration console in SAP HANA Studio
//   + This also gives read access to the metadata of the database
catalog
//   + and read access to the DB configuration
//   * run and analyze authorization traces
//   + this also allows changing any other trace setting

// - Role "security_admin_audit":
//   * Read access to catalog metadata (e.g. table names)
//   + This also allows to enter the administration console
//     in SAP HANA Studio
//   + and read access to the DB configuration
//   * View audit settings
//   * Create, modify, delete audit policies

// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin_extended
  extends role sap-templates.security.protected.roles::security_admin,
  -- which extends sap-
templates.security.protected.roles::security_admin_basic,
  -- and extends
  -- sap-
templates.security.protected.roles::security_admin_troubleshooting,
sap-
templates.security.protected.roles::security_admin_audit
```
5.2.9 Role Security_Admin_Disk_Encryption
This role assembles further privileges that can be useful for a security admin, but are not necessarily essential. Presently, the role allows switching on and off disk encryption (encryption of the data volumes). The required “RESOURCE ADMIN” privilege has the side effect of allowing further data volume administrative tasks. It is also included in ROLE_PERSISTENCE_ADMIN.

5.2.9.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to the metadata of the catalog</td>
</tr>
<tr>
<td>System privilege RESOURCE ADMIN</td>
<td>Here: required to manage disk encryption. Unavoidable side-effect: also allows other tasks on the data volume such as reclaiming free space (“shrinking” the data volume)</td>
</tr>
</tbody>
</table>

5.2.9.2 hdbrole file
The .hdbrole file looks as follows:

```bash
// Role for persons who manages security properties // of the database system. // // This role only contains privileges that not all // security admins may need and that have side-effects // // - Role "security_admin_disk_encryption": //   * Enter administration console in SAP HANA Studio //   + This also gives read access to the metadata of the database catalog //   + and read access to the DB configuration //   * enable/disable disk encryption //   + this also allows "reclaiming" free space in data volume // // There are further security-admin-related privileges collected in // role security_admin_extended (and roles therein), // // Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::security_admin_disk_encryption {
  -- enter into administration console, view system configuration and traces:
  system privilege: CATALOG READ;

  -- enable/disable disk encryption
  -- side-effect: can also re-claim free space in data volume.
  system privilege: RESOURCE ADMIN;
}
```
5.2.10 Role AuditOperator

Defining the security policies and monitoring the security log should generally be separated. If you write audit logs to the syslog daemon, you get the separation by default (only depending on how you configure the syslog itself). If you write audit entries to the trail target “database table”, our proposed security admin roles do not have read or write access to this audit trail. That’s why we introduce the additional role audit_operator.

The role lacks read access to the audit configuration, because today HANA does not offer a privilege to only grant read without write on the audit config.

We also do not add catalog read to the audit operator role, which you might add if you think it is needed.

5.2.10.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege AUDIT OPERATOR</td>
<td>Enable/disable auditing, change audit trail, add, modify or delete audit policies</td>
</tr>
</tbody>
</table>

5.2.10.2 hdbrole file

The .hdbrole file looks as follows:

````
// Role for persons who operates (i.e. views and potentially performs house keeping on the audit trail).
/
/ // Actions permitted by this role:
// - Role "audit_operator":
//   * Read audit entries written to audit trail target database table
//   * Perform housekeeping on audit trail target database table
//     (e.g. delete all entries older than <timestamp>
//     The role does not permit reading the audit settings, and it does not give access to the catalog metadata.
// There are further audit-related privileges collected in role security_admin_audit.
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.protected.roles::audit_operator {
  -- read audit entries in audit trail "database table"
  -- housekeeping on that audit trail
  system privilege: AUDIT OPERATOR;
}
```

5.3 Roles for Database Administrators

In this section we assemble a large number of roles:

- **ROLE SYSTEM_ADMIN GENERIC** which in itself is composed of three further roles:
  - **ROLE BASIC_ADMIN**
  - **ROLE PERSISTENCE_ADMIN**
  - **ROLE BACKUP_ADMIN**
- **ROLE SYSTEM_ADMIN_PREINSTALLED** which includes roles delivered with SAP HANA or add-ons
- **ROLE BACKUP_OPERATOR**
5.3.1 Role basic_admin
This role collects all actions that any DB administrator will expect they are allowed to do and that are not specific to data schemas or repository packages.

5.3.1.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to all metadata of the database catalog. Among other things, required to enter into the administration editor of SAP HANA studio</td>
</tr>
<tr>
<td>System privilege SERVICE ADMIN</td>
<td>Start and stop individual services (processes) of the database</td>
</tr>
<tr>
<td>System privilege INIFILE ADMIN</td>
<td>Modify the database configuration</td>
</tr>
<tr>
<td>System privilege TRACE ADMIN</td>
<td>Start and stop database traces, change the trace levels of the kernel trace</td>
</tr>
<tr>
<td>System privilege SESSION ADMIN</td>
<td>Kill sessions</td>
</tr>
<tr>
<td>System privilege VERSION ADMIN</td>
<td>Trigger garbage collection of the database’s version history (part of MVCC implementation)</td>
</tr>
<tr>
<td>System privilege LICENSE ADMIN</td>
<td>Install or delete license key</td>
</tr>
<tr>
<td>SELECT on schema _SYS_STATISTICS</td>
<td>Read alerts of the statisticsserver process</td>
</tr>
</tbody>
</table>

5.3.1.2 hdbrole file
The .hdbrole file looks as follows:

```
// Role for typical basic database administration tasks
// but not including management of data/log volumes,
// backup management, security management, ...

// Actions permitted by this role:
// - Role "basic_admin":
//   * enter the administrative console of SAP HANA studio (read access)
//   + this also gives read access to the metadata of the database
//   * start and stop DB processes
//   * See alerts
//   * Modify alert settings
//   * Terminate threads and sessions
//   * Make changes to the database configuration (inifiles)
//   * Start and stop traces
//   * stop sessions/threads
//   * trigger garbage collection on version history (MVCC)
//   * install or delete license keys

// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.admin::basic_admin {  
  -- catalog read allows starting the administrative console  
  -- in SAP HANA Studio. It also gives read access to the  
  -- database’s metadata, i.e. we can see the metadata of all
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.3.2 Role persistence_admin
This role enables persistence-related tasks such as cleaning up free (unused) space in data and log volume, or enforcing a savepoint.

5.3.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to all metadata of the database catalog</td>
</tr>
<tr>
<td>System privilege SAVEPOINT ADMIN</td>
<td>Force execution of savepoint</td>
</tr>
<tr>
<td>System privilege RESOURCE ADMIN</td>
<td>Allow freeing up unused space in data volume (shrinking data volume). Side effect: also allows managing disk encryption. If not desired, create your own copy of the persistence_admin role and remove this privilege.</td>
</tr>
<tr>
<td>System privilege LOG ADMIN</td>
<td>Allows freeing up space on log volume (re-claim log space) and switching logging on/off</td>
</tr>
</tbody>
</table>

5.3.2.2 hdbrole file
The .hdbrole file looks as follows:
```
// Role combining privileges needed to administrate
// data and log volumes through the DB layer, i.e.
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.3.3 Role backup_admin

Simple role that allows all backup-related tasks, such as creating a database backup or managing the backup catalog or deleting backups from disk

5.3.3.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read access to all metadata of the database catalog</td>
</tr>
<tr>
<td>System privilege BACKUP ADMIN</td>
<td>Access to all backup functionalities except for restore (which requires OS user credentials)</td>
</tr>
</tbody>
</table>

5.3.3.2 hdbrole file

The .hdbrole file looks as follows:

```
// Role for database user who is allowed to manage database backups:
// Actions enabled by this role:
// - Role "backup_admin":
//   * Enter the backup editor in SAP HANA Studio
//   + this also allows entering the administration console of studio
//   + and it gives read access to the metadata of the database catalog
//   * create backups
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.3.4 Role system_admin_generic
For convenience, we are combining the three roles basic_admin, persistence_admin and backup_admin into one typical generic administrator role.

5.3.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role basic_admin</td>
<td>See 5.3.1</td>
</tr>
<tr>
<td>Role persistence_admin</td>
<td>See 5.3.2</td>
</tr>
<tr>
<td>Role backup_admin</td>
<td>See 5.3.3</td>
</tr>
</tbody>
</table>

5.3.4.2 hdbrole file
The .hdbrole file looks as follows:

```plaintext
// role for the HANA DB system administrator
// this person will need some generic privileges
// to perform typical actions such as changing
// the DB configuration.
// They will also need data specific privileges,
// e.g. in order to trigger delta merges or load
// data into RAM / evict from RAM. Such data specific
// privileges should be added via a second role.
// We have combined groups of administration-related
// privileges into smaller roles, this role here simply
// combines those administrative roles. Depending on
// team setup, you might want to work with the individual
// roles rather than this rather powerful one.
// Actions permitted by this role:
// - Role "basic_admin":
//   * enter the administrative console of SAP HANA studio (read access)
```

```plaintext
role sap-templates.security.common.roles.admin::backup_admin {
  // BACKUP ADMIN allows execution of backups
  // as well as managing backup catalog
  // or deleting backups from system
  system privilege: BACKUP ADMIN;

  // without catalog read, the user cannot open
  // the backup editor in SAP HANA Studio.
  system privilege: CATALOG READ;
}
```
// + this also gives read access to the metadata of the database
catalog
// * start and stop DB processes
// * See alerts
// * Modify alert settings
// * Terminate threads and sessions
// * Make changes to the database configuration (inifiles)
// * Start and stop traces
// * stop sessions/threads

// - Role "persistence_admin":
// * enter the administrative console of SAP HANA studio (read access)
// + this also gives read access to the metadata of the database
catalog
// * force savepoint execution
// * allow "shrinking of data files"
// + this also allows enabling/disabling disk encryption
// * Re-claim log space
// * enable/disable log writing (transaction logs)

// - Role "backup_admin":
// * Enter the backup editor in SAP HANA Studio
// + this also allows entering the administration console of studio
// + and it gives read access to the metadata of the database catalog
// * create backups
// * cancel running backups
// * monitor backup progress
// * delete backups (from disk)
// * read backup catalog
// * manage backup catalog (e.g. delete backup entries)

// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.admin::system_admin_generic
  extends role sap-templates.security.common.roles.admin::persistence_admin,
    sap-templates.security.common.roles.admin::basic_admin,
    sap-templates.security.common.roles.admin::backup_admin
{
  // we do not add individual privileges into this role

  // remember to also give data permissions (update on data schemas)
  // so that the admin can run delta merges and load data into RAM
  // etc.
  // And you might want to add privileges on the repository.
}

## 5.3.5 Role system_admin_preinstalled

SAP HANA comes with a couple if pre-installed roles for administrators, which are typically related to XS-applications coming with the database.

In our role system_admin_preinstalled we collect those SAP-delivered roles that we assume will be useful for a typical DB administrator.

Note that there are other pre-installed roles that are relevant e.g. in a development content, see e.g. our role content_transport_manager in 5.4.4.
5.3.5.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role sap.hana.admin.roles::Monitoring</td>
<td>Access to the new (SPS 7) monitors for memory</td>
</tr>
<tr>
<td></td>
<td>consumption, resource consumption, etc.</td>
</tr>
</tbody>
</table>

5.3.5.2  hdbrole file

The .hdbrole file looks as follows:

```
// Collector for pre-installed roles

// We include here:
// - sap.hana.admin.roles::Monitoring
//   for getting data in the graphical memory- or
//   respource consumption overviews
role sap-templates.security.common.roles.admin::system_admin_preinstalled
  extends role sap.hana.admin.roles::Monitoring
{
  -- nothing to add here
}
```

5.3.6  Role backup_operator

Minimal role for a user who can only create data backups, and do nothing else. This role is de-facto a wrapper around the BACKUP OPERATOR system privilege.

5.3.6.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege BACKUP OPERATOR</td>
<td>Enable creation of data backups</td>
</tr>
</tbody>
</table>

5.3.6.2  hdbrole file

The .hdbrole file looks as follows:

```
// Role for database user who is allowed
// to create database backups, but make no
// other changes to the database

// Actions enabled by this role:
// - Role "backup_operator":
//   * create data backups

// Note that a holder of this role cannot enter into the
// backup editor of SAP HANA studio - that would require
// system privilege BACKUP ADMIN.

// author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.admin::backup_operator {
  // BACKUP OPERATOR allows creation of data backups
  system privilege: BACKUP OPERATOR;
}
```
5.3.7 Role database_monitoring
Role for a read-only user who can perform all typical DB monitoring tasks, such as monitoring memory and other resource consumption, system alerts, view traces, view system configuration, etc.

5.3.7.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read all metadata of the database catalog, including table metadata etc.</td>
</tr>
<tr>
<td>SELECT on schema _SYS_STATISTICS</td>
<td>View alerts from the statisticsserver</td>
</tr>
<tr>
<td>Role sap.hana.admin.roles::Monitoring</td>
<td>Access the new (SPS-7) graphical views for memory and resource consumption</td>
</tr>
</tbody>
</table>

5.3.7.2 hdbrole file
The .hdbrole file looks as follows:

```plaintext
// Role that gives read access to all parts of a HANA DB that are required for typical DB monitoring

// Actions permitted by this role
// - Role database_monitoring
//   * Read access to all tabs of the administration editor of SAP HANA studio
//   * Read access to the alerts of the statisticsserver
//   * Display the new graphical memory- and resource-consumption displays
//   Unfortunately, we cannot include read access to the backup editor in studio
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.admin::database_monitoring
extends role sap.hana.admin.roles::Monitoring
{
  -- read access to all monitoring views,
  -- and ability to open the administration editor of studio
  system privilege: CATALOG READ;

  -- read access to current and historic alerts
  catalog schema "_SYS_STATISTICS": SELECT;
}
```

5.3.8 Role data_admin
This is a fancy role – it defines a user who can

- Create new database schemas directly in the catalog
- Export catalog objects to the DB server (binary or csv)
- Export catalog objects to the client machine (binary or csv)
How To... Define Standard Roles for Administrators in SAP HANA Database

- Import catalog objects from the DB server (binary or csv)
- Import catalog objects from the client machine (binary or csv)

Due to the concept of object ownership, this role defines a very powerful user, because the creator of a DB schema owns the schema and has full access to all objects in the schema.

Similarly, the user who imports catalog objects is the object owner and thus has full access to these objects.

We suggest that this role should only be used in test and development systems, in which developer might need to be able to create their own data objects for trial purposes.

It might be useful in productive systems. However, note that the IMPORT feature basically allows overwriting all catalog objects in the database (IMPORT ... WITH REPLACE).

In our eyes, also the EXPORT privilege should be avoided, because there will normally not be a justification for exporting tables or other objects out of a production system.

And also CREATE SCHEMA should not be needed in production systems, except for initial system setup, e.g. for setting up targets for data services.

5.3.8.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CREATE SCHEMA</td>
<td>Create new schemas directly in the database catalog</td>
</tr>
<tr>
<td>System privilege EXPORT</td>
<td>Export catalog objects to the DB server (csv/binary) or to the client machine</td>
</tr>
<tr>
<td>System privilege IMPORT</td>
<td>Import catalog objects from the DB server (csv/binary) or from the client machine</td>
</tr>
</tbody>
</table>

5.3.8.2 hdbrole file

The .hdbrole file looks as follows:

```
// role for administrative tasks around data management,
// especially for development systems:
//
// The following actions are enabled by this role:
// - role data_admin
//   * create new database schemas
//   * export catalog objects to DB server (binary or csv)
//   * import catalog objects from DB server (binary or csv)
//
// Due to the ownership concept in SAP HANA, this user will
// have full access to all objects in schemas they create,
// or to all objects they import from the server.
//
// Note that exporting catalog objects requires the SELECT
// privilege on the object (or schema containing it).
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.admin::data_admin {  
  // allow creation of schemas:  
  system privilege: CREATE SCHEMA;  
  // allow server-side export:  
  system privilege: EXPORT;  
  // allow import of catalog objects from server:  
  system privilege: IMPORT;  
}  
```
5.4 Roles for Repository Administrators

When building roles for repository administration and development in the repository, one must define first an approach at the general management of the repository and distribution of duties between administrators and developers.

We introduce the following basic rule set which serves as guideline for the roles we build:

- Database administrators have no permissions on the repository
- Repository administrators
  - have no database basis permissions
  - have no edit privileges in the repository, that is they cannot modify individual repository objects
  - can define the package structure of the entire repository
  - can export/import or transport by means of delivery units
  - can define delivery units
- Developers
  - have access only to portions of the repository
  - have edit and activate permissions only on selected packages
  - do not exist in production systems

The roles proposed in this section are intended for people who

- Manage the repository structure, i.e. they may create new packages in the repository
- Read repository content and activate it (e.g. to correct activation failures during imports caused by incorrect dependency resolution during the import)
- Manually export and import repository contents by means of delivery-unit export/import.
- Transport repository contents using the SAP HANA Application Life Cycle Management application sap.hana.xs.lm

We include the following roles here:

- **ROLE_REPO_MANAGER**
- **ROLE_REPO_EXPORTER**
- **ROLE_REPO_IMPORTER**
- **ROLE_CONTENT_TRANSPORT_MANAGER**
- **ROLE_CONTENT_TRANSPORT_EXECUTOR**
- **ROLE_CONTENT_TRANSPORT_SOURCE**

5.4.1 Role repo_manager

The repository manager will be allowed to define the structure of the repository. The repository manager must be aware of company-wide policies regarding the repository structure, for example the boundaries which we propose here, cf. §3.1

The repository manager will be allowed to create or delete packages in the repository, and they will be empowered to activate objects (e.g. for the case that automatic activation during import fails).

⚠️ Note

Even though the repo manager has the REPO.ACTIVATE-privileges, they may not have all privileges required to activate certain content. Depending on the type of content to be activated, they might require additional system or object privileges.
5.4.1.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
</tbody>
</table>
| package .REPO_PACKAGE_ROOT: REPO.READ, REPO.MAINTAIN_NATIVE_PACKAGES, REPO.ACTIVATE_NATIVE_OBJECTS, REPO.ACTIVATE_IMPORTED_OBJECTS | Permission on the entire repository to  
- Read repo contents  
- Create new packages  
- Activate new and imported objects |
| System privilege | Create, modify and delete delivery unit definitions |

5.4.1.2 hdbrole file

The .hdbrole file looks as follows:

```sql
// Role for person that manages the SAP HANA Repository  
// The following actions are enabled by this role:  
// - Role repo_manager  
//   * general access to the repository (repository_rest)  
//   * permission to manager repository packages anywhere in the repository  
//   * activate native and imported objects in the entire repository  
//   * Maintain (add/modify/delete) delivery units  
role sap-templates.security.common.roles.repository::repo_manager {  
  -- allow access to the repository at all:  
  catalog sql object "SYS"."REPOSITORY_REST": EXECUTE;  
  -- Manage repository contents (in the full repository):  
  -- Read all repository contents  
  package .REPO_PACKAGE_ROOT: REPO.READ;  
  -- Read create new packages (within locally created packages)  
  package .REPO_PACKAGE_ROOT: REPO.MAINTAIN_NATIVE_PACKAGES;  
  -- activate native objects  
  package .REPO_PACKAGE_ROOT: REPO.ACTIVATE_NATIVE_OBJECTS;  
  -- activate imported objects (from DU imports)  
  package .REPO_PACKAGE_ROOT: REPO.ACTIVATE_IMPORTED_OBJECTS;  
  -- allow management of delivery units (create/delete DUs)  
  system privilege: REPO.MAINTAIN_DELIVERY_UNITS;  
}
```

5.4.2 Role repo_exporter

This role allows to  
- Define, modify and delete delivery units  
- Export delivery units to the server or client

⚠️ CAUTION

This role allows exporting all contents of the repository, to the file system on the database server, and also to the client computer.
Delivery units can only be exported if the exporting user has read permission to the packages attached to the delivery unit. For this general purpose role, we grant read access on the entire repository, hence the above warning.

### 5.4.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>package.REPO_PACKAGE_ROOT: REPO.READ</td>
<td>Full read access to the entire repository. This is required to enable the exporting of packages.</td>
</tr>
<tr>
<td>System privilege REPO.MAINTAIN_DELIVERY_UNITS</td>
<td>Create, modify and delete delivery unit definitions</td>
</tr>
<tr>
<td>System privilege REPO.EXPORT</td>
<td>Export delivery units to server or client side</td>
</tr>
<tr>
<td>package.REPO_PACKAGE_ROOT: REPO.MAINTAIN_NATIVE_PACKAGES</td>
<td>Allow attaching native packages to delivery units</td>
</tr>
<tr>
<td>package.REPO_PACKAGE_ROOT: REPO.MAINTAIN_IMPORTED_PACKAGES</td>
<td>Allow attaching imported packages to delivery units</td>
</tr>
</tbody>
</table>

### 5.4.2.2 hdbrole file

The .hdbrole file looks as follows:

```sql
// Role that allows manual exporting of repository contents by means of delivery-unit exports.
//
// Actions enabled by this role:
// - Role "repo_exporter":
//   * Generic access to the repository
//   * creation and deletion of delivery units
//   * manual export of delivery units to server or client
//   * read access to the entire repository - to allow exporting arbitrary content if needed
//   * adding native and imported packages to DUs/
// Note that by importing a delivery unit all contents of the packages therein can be overwritten. Hence you must control usage of the export and import roles strictly
//
// For general tasks of life cycle management, you should make use of application life cycle management tools such as CTS+ or the HANA application life cycle management application sap.hana.xs.lm which is part of SAP HANA SPS 6 or higher.
//
// We suggest that this role is only used for support purposes, but not for regular management of repository content.
//
// author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.repository::repo_exporter {
  -- generic access to the repository
  catalog sql object "PUBLIC"."REPOSITORY_REST": EXECUTE;
  -- allow exporting delivery units manually
  system privilege: REPO.EXPORT;
  -- allow management of delivery units (create/delete DUs)
  system privilege: REPO.MAINTAIN_DELIVERY_UNITS;
}
--- exporting of delivery units is only possible, if the
--- exporting user has read access to the packages attached
--- to the delivery unit.
--- Since this role is intended mainly for emergency
--- purposes (in production), we include read access
--- to the entire repository here.
package .REPO_PACKAGE_ROOT: REPO.READ;
--- Defining delivery unit (attaching package) requires the
--- maintain package privilege
--- for native packages
package .REPO_PACKAGE_ROOT: REPO.MAINTAIN_NATIVE_PACKAGES;
--- and for imported packages
package .REPO_PACKAGE_ROOT: REPO.MAINTAIN_IMPORTED_PACKAGES;
}

5.4.3 Role repo_importer
This role allows manual importing of delivery units. This is necessary, for example, for implementing applications or other content that is delivered as DU exports, such as the SHINE demo package.

⚠️ CAUTION
Importing delivery units cannot be restricted to certain areas of the repository. A holder of this role thus can de-facto overwrite all repository contents.

⚠️ CAUTION
Please take special note that we also cannot distinguish between client-side and server-side DU imports. Hence the holder of this role may fabricate a delivery unit on the client side and import it into the system.

⚠️ Tip
You might want to make sure that holder of this role have no read access to the repository whatsoever. This means especially that no database user (and no natural person) should have the repo_importer and the repo_exporter roles at the same time.

⚠️ Tip
For the purpose of application life cycle management please use the provided tools, i.e. either the application delivered with SAP HANA (sap.hana.xs.lm) or SAP’s extended transport system CTS+. Avoid manual export/import.

We suggest not to use this role in production systems.

5.4.3.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository</td>
</tr>
<tr>
<td>System privilege REPO.IMPORT</td>
<td>Import delivery units</td>
</tr>
</tbody>
</table>
5.4.3.2  hdbrole file

The .hdbrole file looks as follows:

```bash
// Role that allows manual import of repository contents
// by means of delivery-unit imports
/
/
//   Actions enabled by this role:
//   - Role "repo_importer":
//     * general access to the repository
//     * manual import of delivery units from server or client
//   Note that by importing a delivery unit all contents of the
// repository can technically be overwritten.
//   For general tasks of life cycle management, you should make
// use of application life cycle management tools such as CTS+
// or the HANA application life cycle management application
// sap.hana.xs.lm which is part of SAP HANA SPS 6 or higher.
// We suggest using this role only in exceptional circumstances,
// e.g. when you need to manually load a delivery unit. This
// might be the case when you implement an externally provided
// content package such as the SHINE demo content for SAP HANA.
// author: Richard Bremer <richard.bremer@sap.com>

role sap-templates.security.common.roles.repository::repo_importer {
  -- generic access to the repository
  catalog sql object "PUBLIC"."REPOSITORY_REST": EXECUTE;
  -- allow importing delivery units manually
  system privilege: REPO.IMPORT;
}
```

5.4.4  Role content_transport_manager

Role required for a person who wants to transport HANA repository contents using the built-in SAP
HANA Application Lifecycle Management application which we like to call HALMA.
This role is simply the collection of all roles required to set up and execute transports, as described
in the SAP HANA developer guide.

⚠️ **CAUTION**

Please note that the roles delivered with the sap.hana.xs.lm application are extremely
powerful. The role sap.hana.xs.lm.roles::Administrator, for example, contains the
following privileges (among others):
System privilege  INIFILE_ADMIN → modify the database configuration
Package Privileges REPO.EDIT_NATIVE_OBJECTS and
REPO.EDIT_IMPORTED_OBJECTS on .REPO_PACKAGE_ROOT → edit all objects
(imported or not) in the repository
The predelivered roles will be adjusted in the next HANA releases. Until then, we
propose to only grant the role during the phase of system setup and revoke it once the
transport system is established. Use role “content_transport_executor” from then on.

You might want to separate transport management into users that can define transports and users
that can execute transports. We decided to include the ability to execute transports in the transport
manage role because of the above suggestion to only use it during the setup phase. Once the transport system is set up and tested, you should only work with users that have the transport executor role.

### 5.4.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role sap.hana.xs.lm.roles::Administrator</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
<tr>
<td>Role EXECUTE_TRANSPORT in package sap.hana.xs.lm.roles</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
<tr>
<td>Role HTTPDestAdministrator in package sap.hana.xs.admin.roles::</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
<tr>
<td>Role RuntimeConfAdministrator in package sap.hana.xs.admin.roles</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
</tbody>
</table>

### 5.4.4.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Role for user that may set up system connections, // transport routes and execute transports // using the SAP HANA Application Lifecycle // Management Application (HALMA) // // The role simply combines all SAP-delivered roles // that are required for this task. // // The following actions are enabled by this role: // - Role content_transport_manager //   * define HTTP destinations (register systems) //   * define transport routes //   * Execute transports // // There are most probably side-effects from further // actions enabled by the SAP-delivered roles we are // using here. For details, refer to the role descriptions // in SAP's official product documentation. // // Author: Richard Bremer <richard bremer@sap.com>
role sap-templates.security.common.roles.repository::content_transport_manager
extends role sap.hana.xs.lm.roles::Administrator, sap.hana.xs.lm.roles::ExecuteTransport, sap.hana.xs.admin.roles::HTTPDestAdministrator, sap.hana.xs.admin.roles::RuntimeConfAdministrator
{
    // nothing to add here
}
```
5.4.5  Role content_transport_executor
Role required for a person who wants to transport HANA repository contents using the built-in SAP HANA Application Lifecycle Management application which we like to call HALMA. This role does not allow the definition of transports (setting up system connections or transport routes). See our role content_transport_manager for that purpose.

5.4.5.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role EXECUTE_TRANSPORT in package sap.hana.xs.lm.roles</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
</tbody>
</table>

5.4.6  hdbrole file
The .hdbrole file looks as follows:

```java
// Role for user that may set up system connections, // transport routes and execute transports // using the SAP HANA Application Lifecycle // Management Application (HALMA) // // The role simply combines all SAP-delivered roles // that are required for this task. // // The following actions are enabled by this role: // * Role content_transport_executor // * Execute transports // // There are most probably side-effects from further // actions enabled by the SAP-delivered roles we are // using here. For details, refer to the role descriptions // in SAP's official product documentation. // // Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.repository::content_transport_executor extends role sap.hana.xs.lm.roles:: ExecuteTransport {
    // nothing to add here
}
```

5.4.6  Role content_transport_source
Role required for providing the transport container in the source system of a transport. We simply provide here a wrapper for the SAP-delivered role from the sap.hana.xs.lm application

5.4.6.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role TRANSPORT in package sap.hana.xs.lm.roles</td>
<td>See SAP documentation, e.g. SAP HANA developer guide</td>
</tr>
</tbody>
</table>
5.4.6.2  **hdbrole file**

The .hdbrole file looks as follows:

```java
// Role for user that may set up system connections, transport routes and execute transports
// using the SAP HANA Application Lifecycle Management Application (HALMA)
// The role simply combines all SAP-delivered roles that are required for this task.
// The following actions are enabled by this role:
//  - Role content_transport_source
//    * Execute transports
//    There are most probably side-effects from further actions enabled by the SAP-delivered roles we are using here. For details, refer to the role descriptions in SAP's official product documentation.
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.repository::content_transport_source
extends role sap.hana.xs.lm.roles::Transport
{
  // nothing to add here
}
```

5.5  **Roles for Developers**

It is not possible for us to provide all roles for developers in customer landscapes. One reason is that developers might need read or even write access to database schemas – and we cannot know what schemas a customer is going to have.

However, we can provide several roles that have all generic privileges required for developing, i.e. those that are not related to objects, packages or applications (others than official SAP-content-applications).

On the repository-side of things, because of our proposed public space for free-style testing (package system-local.public) we can always show you what package privileges are required by including them for package system-local.public. If your actual system, you will have to add (roles with) privileges on your project-/team-specific packages in system-local.private or <vendor>.

We have divided the roles into two parts: one for generic development in SAP HANA, such as development for the XS engine. For that topic, we only include two rather generic roles.

The second area is data modeling, see 5.6.

We include the following roles here:

- **ROLE XS_APP_DEV GENERIC**
  Contains system privileges and other generic privileges for general development, but no privileges on specific packages or catalog objects (data, ...)

- **ROLE XS_APP_DEV_PUBLIC**
  Example role with full developer rights for XS app development on package system-local.public; does not contain any privileges on catalog objects
5.5.1  Role xs_app_dev_generic

It’s almost impossible to build a generic development role, because the really interesting bit are the package and data specific privileges.

This role contains everything except these specific privileges. See ROLE XS_APP_DEV_PUBLIC for a role that includes package privileges.

You can give this role to any developer, and add the appropriate privileges on packages and catalog-objects via further dedicated roles that you will need to create in your development environment.

5.5.1.1  Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Full read access to all metadata of the database catalog, e.g. list of all tables, definitions of all tables, … without giving any data access</td>
</tr>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository (no access to any content objects)</td>
</tr>
</tbody>
</table>

5.5.1.2  hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Generic privileges needed for anyone who wants to create // applications on SAP HANA XS // // This role contains the generic requirements for application development, // i.e. those privileges that are independent from customer-specific // objects such as catalog schemas/tables, repository packages, etc. // // You can easily create your own developer roles for development // projects in package <x> by granting the same package privileges // on package <x>, and adding access to catalog objects as required. // // The following actions are enabled by this role: // - Role xs_app_dev_generic // + read access to the metadata of the database catalog // + general access to the repository (REPOSITORY_REST) // // author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.development::xs_app_dev_generic {  // System Privileges  // Allow the user to verify that the tables underlying the // development project do exist.  // We simply grant CATALOG READ, so the user has no SELECT  // Privilege on the underlying database tables.  // This will allow reading _all_ catalog metadata in the system,  // but no table contents.  system privilege: CATALOG READ;  // Here we allow the role to read the repository tree (expand the  // "content" tree). This does not allow seeing the content of packages  catalog sql object "SYS"."REPOSITORY_REST": EXECUTE;  // You will have to add the following privileges for a development  // project working in package <project_package>
```
5.5.2 Role xs_app_dev_public
Developer role for free-style development of XS apps in package system-local.public. The role does not contain any access to catalog objects, but does include read access to the metadata of the database catalog.

You will probably need to set up further roles giving developers access to catalog objects, specific to your environment.

5.5.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Full read access to all metadata of the database catalog, e.g. list of all tables, definitions of all tables, … without giving any data access</td>
</tr>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository (no access to any content objects)</td>
</tr>
<tr>
<td>package system-local.public: REPO.READ, REPO.MAINTAIN_NATIVE_PACKAGES, REPO.EDIT_NATIVE_OBJECTS, REPO.ACTIVATE_NATIVE_OBJECTS</td>
<td>Public package for free-style testing system-local.public: Full access to repository contents, including right to create new packages and to create, edit and activate native objects. Also read access to the definition of any native or imported object</td>
</tr>
</tbody>
</table>

5.5.2.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Generic privileges needed for anyone who wants to create applications on SAP HANA XS
// This role contains everything required application development in package system-local.public
// except for any data access (catalog object access).
// You can easily create your own developer roles for development projects in package <x> by granting the same package privileges on package <x>, and adding access to catalog objects as required.

// The following actions are enabled by this role:
// - Role xs_app_dev_generic
// + read access to the metadata of the database catalog
// + general access to the repository (REPOSITORY_REST)
// + full read access to package system-local.public
```
5.6 Roles for Data Modelers

We propose the following roles for data modeling:

- **ROLE DATA MODELING GENERIC**
  Contains all privileges required for creation of data models, including required package privileges for package system-local.public but no rights on any customer-specific catalog objects and also no permissions for previewing any data models.

- **ROLE DATA MODELING PUBLIC**
  Contains all privileges from role data_modeling_generic, plus package privileges on the public development package system-local.public – i.e. most privileges required for development in the public development space. The role still is not complete, because one also needs to add privileges on the catalog objects underlying the data models. These catalog objects are specific to the development project and thus cannot be contained in a general purpose role.

- **ROLE MODEL PREVIEW UNRESTRICTED**
  Contains all privileges required to read from activated data models, without any restrictions.

- **ROLE DATA MODELING**
  A combination of the above roles data_modeling_public and model_preview_unrestricted – for a typical developer who can build and test data models.
• **ROLE BUILD ANALYTIC PRIVILEG**
  Contains all privileges required to develop analytic privileges in package system-local.public
  It is therefore a test role for free-style testing with analytic privileges
  Please note our proposed role build_analytic_privilege_generic in the security area 5.7.1.

### 5.6.1 Role data_modeling_generic

This role contains all privileges required to create and activate data models, except for the required read access to customer-specific catalog objects and except for package privileges.

We do include here:
- Any system privileges required
- Any generic catalog privileges required
- SELECT access to schema _SYS_BI which contains e.g. system-generated time-dimension tables

In addition, developers will need package privileges on the development package for their project.

See our proposed **ROLE DATA MODELING PUBLIC**.

In order to empower developers to create data models on top of data in schemas <x> and <y>, you will have to give them a role granting the SELECT (and, depending on the scenario) the EXECUTE privilege on schemas <x> and <y>.

Note that this is a minimal developer role that also does not include any privilege to read from activated data models.

In most cases, you will want to give your developers also the **ROLE MODEL_PREVIEW_UNRESTRICTED** – at least as long as they are not working on sensitive data.

#### 5.6.1.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Full read access to all metadata of the database catalog, e.g. list of all tables, definitions of all tables, ... without giving any data access</td>
</tr>
<tr>
<td>System privilege CREATE SCENARIO</td>
<td>Dedicated system privilege to enable activation of data models</td>
</tr>
<tr>
<td>System privilege CREATE R SCRIPT</td>
<td>Develop R script on HANA</td>
</tr>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository (no access to any content objects)</td>
</tr>
<tr>
<td>SELECT on schema _SYS_BI</td>
<td>Read data in schema _SYS_BI, especially table M_TIME_DIMENSION which contains system-generated timestamp mapping/conversion information</td>
</tr>
</tbody>
</table>

#### 5.6.1.2 hdbrole file

The .hdbrole file looks as follows:
```
// Generic privileges needed for anyone who wants to do data modeling
// This role contains everything required for modeling
// except for:
```
How To... Define Standard Roles for Administrators in SAP HANA Database

// - package privileges
// - privileges on application data schemas
// we also do not include here the privileges required for
// activating analytic privileges - see role
// build_analytic_privileges_generic
// we also don't include privileges for reporting/data preview,
// see role model_preview_unrestricted

The following actions are enabled by this role:
// - Role data_modeling_generic
//   * read access to all metadata of the db catalog
//   * creation of R scripts
//   * activation of data models (CREATE SCENARIO)
//   * generic access to the repository (REPOSITORY_REST)
//   * read access to schema _SYS_BI which contains
//     system-generated time dimension tables as well
//     as model metadata, but no application data

This role must always be accompanied by the complementary
roles including data schema access and package privileges
// to enable data modeling.

author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.modeling::data_modeling_generic {
  // System Privileges:
  // Allow the user to verify that the tables underlying the Information
  // Models do exist. We simply grant CATALOG READ, so the user has no
  // SELECT Privilege on the underlying database tables. This will
  // allow reading _all_ metadata in the system, but no table contents.
  system privilege: CATALOG READ;
  // Here we also allow R Script to be created by this role
  system privilege: CREATE R SCRIPT;
  // For activation of data models, you need CREATE SCENARIO:
  system privilege: CREATE SCENARIO;

  // If you want to, you could add authorization to
  // create analytic privileges
  system privilege: CREATE STRUCTURED PRIVILEGE;
  // and to modify/re-activate analytic privileges:
  system privilege: STRUCTUREDPRIVILEGE ADMIN;

  // Here we allow the role to read the repository tree
  // (expand the "content" tree). This does not allow seeing
  // the content of packages
  catalog sql object "SYS"."REPOSITORY_REST" EXECUTE;

  // We also enable activation of time-based Attribute Views as these are
  // based on a standard table in schema _SYS_BI
  // for simplicity, we include the entire _SYS_BI schema which does not
  // contain sensitive data
  catalog schema ".SYS_BI" SELECT;

  // data access
  // Strictly speaking, no SELECT or EXECUTE privileges are required in
  // order to build and activate data models, as long as the modeler has
  // access to the catalog metadata (system privilege CATALOG READ).
  // It is, however, easier to perform data modeling tasks if the modelers
// have SELECT access to the database schemas underlying the data models.
// repository access
// modelers will require all "native" package privilege on the package
// in which they want to work, including REPO.READ.
// Just like the data access, these privileges should be placed into
// a dedicated role for the project.

5.6.2 Role data_modeling_public
This role contains all privileges required to create and activate data models, except for the required read access to customer-specific catalog objects and except for package privileges.

We do include here:

- Role data_modeling_generic which provides:
  - Any system privileges required
  - Any generic catalog privileges required
  - SELECT access to schema _SYS_BI which contains e.g. system-generated time-dimension tables

- All required privileges to do data modeling on package system-local.public

In addition, developers will need package privileges on the development package for their project.

See our proposed

In order to empower developers to create data models on top of data in schemas <x> and <y>, you will have to give them a role granting the SELECT (and, depending on the scenario) the EXECUTE privilege on schemas <x> and <y>.

Note that this is a minimal developer role that also does not include any privilege to read from activated data models.

In most cases, you will want to give your developers also the ROLE MODEL_PREVIEW_UNRESTRICTED – at least as long as they are not working on sensitive data.

5.6.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Full read access to all metadata of the database catalog, e.g. list of all tables, definitions of all tables, ... without giving any data access</td>
</tr>
<tr>
<td>System privilege CREATE SCENARIO</td>
<td>Dedicated system privilege to enable activation of data models</td>
</tr>
<tr>
<td>System privilege CREATE R SCRIPT</td>
<td>Develop R script on HANA</td>
</tr>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository (no access to any content objects)</td>
</tr>
<tr>
<td>SELECT on schema _SYS_BI</td>
<td>Read data in schema _SYS_BI, especially table M_TIME_DIMENSION which contains system-generated timestamp mapping/conversion information</td>
</tr>
<tr>
<td>Privilege</td>
<td>What does it do?</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>package system-local.public: REPO.READ, REPO.EDIT_NATIVE_OBJECTS, REPO.ACTIVATE_NATIVE_OBJECTS, REPO.MAINTAIN_NATIVE_PACKAGES</td>
<td>Public package for free-style testing system-local.public: Full access to repository contents, including right to create new packages and to create, edit and activate native objects. Also read access to the definition of any native or imported object</td>
</tr>
</tbody>
</table>

### 5.6.2.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Privileges needed for anyone who wants
// to do data modeling in package system-local.public
//
// This role contains everything required for modeling
// except for:
// - privileges on application data schemas
// we also do not include here the privileges required for
// activating analytic privileges - see role
// build_analytic_privileges_generic
//
// we also don't include privileges for reporting/data preview,
// see role model_preview_unrestricted
//
// The following actions are enabled by this role:
// - Role data_modeling_generic
//  * read access to all metadata of the db catalog
//  * creation of R scripts
//  * activation of data models (CREATE SCENARIO)
//  * generic access to the repository (REPOSITORY_REST)
//  * read access to schema _SYS_BI which contains
//   system-generated time dimension tables as well
// as model metadata, but no application data
// - Role data_modeling_public
//  * privileges on package system-local.public:
//   + REPO.READ - read model definitions
//   + REPO.EDIT_NATIVE_OBJECTS - create and edit objects
//   + REPO.ACTIVATE_NATIVE_OBJECTS
//   + REPO.MAINTAIN_NATIVE_PACKAGES - create packages
//
// This role must always be accompanied by the complementary
// roles including data schema access
// to enable data modeling.
//
// author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.modeling::data_modeling_public
  extends role
    sap-templates.security.common.roles.modeling::data_modeling_generic
{
  // repository access
  // all "native" package privilege on package system-local.public
  // including REPO.READ
  package system-local.public: REPO.READ;
  package system-local.public: REPO.EDIT_NATIVE_OBJECTS;
  package system-local.public: REPO.ACTIVATE_NATIVE_OBJECTS;
  package system-local.public: REPO.MAINTAIN_NATIVE_PACKAGES;
```
How To... Define Standard Roles for Administrators in SAP HANA Database

5.6.3 Role model_preview_unrestricted

Our roles for data modeling do not give read access to the contents of data models. In most cases, a modeler/developer will get unrestricted read access to their data models.

This role model_preview_unrestricted allows

- Running SQL queries against all activated data models to retrieve unrestricted data
- Using BI clients such as Analysis for Office, BusinessObjects Explorer, Lumira, ... or MS Excel to retrieve unrestricted data from activated data models.

In combination with a developer/modeler role which contains package privileges on <package>, it allows data preview in HANA studio on activated models in <package>.

5.6.3.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General repository access – in fact, we assume you have developer rights anyway, so this is not really needed</td>
</tr>
<tr>
<td>SELECT on schema _SYS_BI</td>
<td>Data Preview and many reporting applications require model metadata from objects like BIMC_CUBES in this schema</td>
</tr>
<tr>
<td>SELECT on schema _SYS_BIC</td>
<td>Full object-level access to all activated data models</td>
</tr>
<tr>
<td>EXECUTE on schema _SYS_BIC</td>
<td>Full object-level access to all activated procedures in the default schema for data models</td>
</tr>
<tr>
<td>Analytic privilege _SYS_BI_CP_ALL</td>
<td>“SAP ALL” analytic privilege: unrestricted access to the content of all activated data models</td>
</tr>
</tbody>
</table>

5.6.3.2 hdbrole file

The .hdbrole file looks as follows:

```bash
// Role containing the generically required privileges for testing data models, including activated stored procedures.
// We give:
// Actions enabled by this role
// - Role model_preview_unrestricted
//   * full select and execute on schema _SYS_BIC
//   * full select on schema _SYS_BI
//   * the "see all" analytic privilege _SYS_BI_CP_ALL
//   * execute on the repository stored procedure REPOSITORY_REST
// -> With this combination, the holder of this role has
```
// full read access to all activated data models and
// can execute all activated procedures in the system.
// note that data preview itself is not possible, because
// of a lack of package
// privileges. This typically comes with a scenario-specific
// role for modelers which should include SELECT on the data
// schema underlying the models, as well as access to some
// package(s). Hence a modeler who has a role for modeling
// as well as this role for data preview will be able
// also do actual data preview from within studio.
// author: Richard Bremer <richard.bremer@sap.com>
role sap-
templates.security.common.roles.modeling::model_preview_unrestricted{
  // access to the repository stored procedure
  // REPOSITORY_REST to enable data preview in SAP HANA Studio
  // if the user is also given repository package access
  // (REPO.READ will be sufficient). We do not include
  // package-specific privileges in this generic role.
  catalog sql object "SYS"."REPOSITORY_REST": EXECUTE;
  // If modelers shall be able to use front-ends such as
  // Analysis for Office, SAP BusinessObjects Explorer or
  // MS Excel for checking their data models, then they
  // will also need SELECT privileges on the
  // BIMC-tables/views in schema _SYS_BI.
  // Here we grant SELECT on the entire schema, although
  // this is strictly a little bit more than what is required.
  catalog schema "SYS_BI": SELECT;
  // To allow modelers to activate and read their activated
  // models they need Select on the activated object in
  // schema _SYS_BIC.
  // For simplicity, we grant SELECT on the entire schema,
  // you can of course break this down into individual
  // objects if required.
  catalog schema "SYS_BIC": SELECT;
  // For testing activated procedures,
  // the modeler will need execute on that procedure runtime
  // object (you may want to remove this privilege or confine
  // it to individual procedures, depending on security
  // requirements).
  catalog schema "SYS_BIC": EXECUTE;

  // Analytic Privileges:
  // Include Analytic Privileges using the keywords
  // "analytic privilege" or "catalog analytic privilege"
  // (depending on how the AP has been created).
  // Here we grant the "SAP_ALL" of the analytic privilege world
  // Note that _SYS_BI_CP_ALL allows reading the content of
  // all activated data models (in combination with
  // SELECT on schema _SYS_BIC).
  catalog analytic privilege: "SYS_BI_CP_ALL";
}
5.6.4 Role data_modeling
This role is a typical developer role for data modelers. It contains everything needed, except for package privileges on project-specific packages, because we cannot know the repository layout chosen for a given project.

The role enables:
- Creating and activating data models in package system-local.public
- Previewing any activated data model in the entire system with full data access

Project specific extensions to the role:
- Optional but convenient: SELECT and potentially EXECUTE privileges on database schemas underlying the data model
- Package privileges on project-specific development packages

5.6.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role data_modeling_public</td>
<td>See 5.6.2</td>
</tr>
<tr>
<td>Role model_preview_unrestricted</td>
<td>See 5.6.3</td>
</tr>
</tbody>
</table>

5.6.4.2 hdbrole file
The .hdbrole file looks as follows:

```sql
// Privileges needed for anyone who wants to do data modeling in package system-local.public
// and to preview any activated data model in the database system.
//
// This role contains everything required for modeling except for:
// - privileges on application data schemas
//
// we also do not include here the privileges required for activating analytic privileges - see role build_analytic_privileges_generic
//
// The following actions are enabled by this role:
// - Role data_modeling_generic
//   * read access to all metadata of the db catalog
//   * creation of R scripts
//   * activation of data models (CREATE SCENARIO)
//   * generic access to the repository (REPOSITORY_REST)
//   * read access to schema _SYS_BI which contains system-generated time dimension tables as well as model metadata, but no application data
// - Role data_modeling_public
//   * privileges on package system-local.public:
//     + REPO.READ - read model definitions
//     + REPO.EDIT_NATIVE_OBJECTS - create and edit objects
//     + REPO.ACTIVATE_NATIVE_OBJECTS
//     + REPO.MAINTAIN_NATIVE_PACKAGES - create packages
// - Role model_preview_unrestricted
//   * full select and execute on schema _SYS_BIC
```
5.6.5 Role build_analytic_privilege_public

Example role that allows the creation of analytic privileges in our public free-style test package system-local.public, referring to data models in that same package.

The role inherits all generic (package- and data-independent) privileges from our protected security role sap-templates.security.protected.roles::build_analytic_privilege_generic (see 5.7.1).

For building privileges with static restrictions, you might want to add read access to data models, so that the value help can be used.

The privileges of this role are sufficient to build dynamic analytic privileges, including repository-based definition of stored procedure and table for the lookup of restriction values.

Maintaining the restriction values for dynamic analytic privileges in the lookup-table will need write access to the table (we believe you might even be able to maintain those values via csv import out of the repository, without additional privileges, but we are not certain and did not test, because this procedure would not make sense in a security context).

5.6.5.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role build_analytic_privilege_generic</td>
<td>All privileges required for building analytic privileges, except for package privileges (and maybe data access)</td>
</tr>
<tr>
<td>package system-local.public: REPO.READ,</td>
<td>Public package for free-style testing system-local.public: Full access to repository contents, including right to create new packages and to create, edit and activate native objects. Also read access to the definition of any native or imported object</td>
</tr>
<tr>
<td>REPO.EDIT_NATIVE_OBJECTS, REPO.ACTIVATE_NATIVE_OBJECTS, REPO.MAINTAIN_NATIVE_PACKAGES</td>
<td></td>
</tr>
</tbody>
</table>

5.6.5.2 hdbrole file

The .hdbrole file looks as follows:

```sql
// role containing anything generically required
// for building and activating analytic privileges
// in package system-local.public, based on data models
// in that same package.
```
It is thus intended for developers to test and play with analytic privileges. Everything except for package privileges is inherited from role build_analytic_privilege_generic. The role does not contain any data preview capabilities, so you cannot pick attribute values from the value help. If acceptable, you might grant our data_preview_unrestricted role to developers of analytic privileges.

The following actions are enabled by this role:
- Role build_analytic_privilege_generic
  * General access to the repository (REPOSITORY_REST)
  * Initial activation of analytic privileges
  * Re-activation of analytic privileges
- Role system-local_build_analytic_privilege
  * developer privileges on package system-local.public:
    + REPO.READ - read model definitions
    + REPO.EDIT_NATIVE_OBJECTS - create and edit objects
    + REPO.ACTIVATE_NATIVE_OBJECTS
    + REPO.MAINTAIN_NATIVE_PACKAGES - create packages

5.7 Roles for security developers

At present, we have only one role here, the role with the generic bits required to create analytic privileges. Of course, a role builder is also a security developer, but we felt like having a dedicated section just for role builders – for example because this is the first role you will typically need in your system, in order to define further roles for the users you create.

5.7.1 Role build_analytic_privilege_generic

Generic role fragment for the creation of analytic privileges. This role contains anything required for this task, except for package- and data-specific privileges.

You will always need to extend this role with further privileges. Assuming you create analytic privileges in package <priv_package> and these privilege refer to (give read access to the content of) data models in package <model_package>, you will need:
• `<priv_package>`:  
  REPO.READ, REPO.EDIT_NATIVEOBJECTS, REPO.ACTIVATE_NATIVEOBJECTS and potentially REPO.MAINTAIN_NATIVE_PACKAGES

• `<model_package>`:  
  REPO.READ

For building privileges with static restrictions, you might want to add read access to data models, so that the value help can be used. These might, e.g. come from our role data_preview_unrestricted.

### 5.7.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CREATE STRUCTURED PRIVILEGE</td>
<td>Dedicated system privilege to enable the initial activation of analytic privileges</td>
</tr>
<tr>
<td>System privilege STRUCTUREDPRIVILEGE ADMIN</td>
<td>Dedicated system privilege to enable re-activation of analytic privileges</td>
</tr>
<tr>
<td>EXECUTE on REPOSITORY_REST</td>
<td>General access to the repository (no access to any content objects)</td>
</tr>
</tbody>
</table>

### 5.7.2 hdbrole file

The .hdbrole file looks as follows:

```
// role containing anything generically required
// for building and activating analytic privileges
// except for package privileges

// the role does not contain any data preview capabilities,
// so you cannot pick attribute values from the value help.
// If acceptable, you might grant our data_preview_unrestricted
// role to developers of analytic privileges.

// The following actions are enabled by this role:
// - Role build_analytic_privilege_generic
//   * General access to the repository (REPOSITORY_REST)
//   * Initial activation of analytic privileges
//   * Re-activation of analytic privileges
// - You will always have to extend this role by:
//   - developer privileges on package in which privileges will be created
//   - REPO.READ on packages containing data models to which privileges will refer.
// - Those privileges are project-specific and should be added via dedicated roles.
// author: Richard Bremer <richard.bremer@sap.com>

role sap-templates.security.protected.roles::build_analytic_privilege_generic {
  -- Here we allow accessing (expanding) the "content" tree.
  -- This does not allow seeing the content of packages
  catalog sql object "SYS"."REPOSITORY_REST": EXECUTE;
  -- initial creation of AP:
```
system privilege: CREATE STRUCTURED PRIVILEGE;
-- modification (re-activation) of APs.
system privilege: STRUCTUREDPRIVILEGE ADMIN;

-- you will have to add:
-- repository access
-- all "native" package privilege on package <priv_package> in which you
-- build privileges, including REPO.READ
-- package <priv_package>: REPO.READ;
-- package <priv_package>: REPO.EDIT_NATIVE_OBJECTS;
-- package <priv_package>: REPO.ACTIVATE_NATIVE_OBJECTS;
-- package <priv_package>: REPO.MAINTAIN_NATIVE_PACKAGES;

-- and read access to package(s) <model_package> containing the models
-- your privileges refer to:
-- package <model_package>: REPO_READ;

-- and potentially read access to the data models to look up static
-- restriction values (which typically won't make sense on non-productive
-- data).

-- when building dynamic analytic privileges, you might use
-- repository-defined procedures and tables for restriction value
-- lookup. We assume that these procedure and table definitions are
-- also placed underneath <priv_package>. Otherwise, you will have
-- to include developer rights also on those packages.

5.8 Roles for End-Users
We only have one role template here. It is a role containing generically required privileges for end-users that report against data models.

5.8.1 Role reporting_generic
Our generic reporting role suffers from the fact that most privileges required for reporting are specific to the actual project. This is why we only grant a single privilege here, but we list the other privileges you typically need for reporting here as well.

5.8.1.1 Granted privileges:

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT on schema _SYS_BI</td>
<td>Most BI tools require access to the model-metadata objects such as BIMC_CUBES in schema _SYS_BI. This schema does not contain application data, unless you’re crazy.</td>
</tr>
</tbody>
</table>

5.8.1.2 Further privileges needed for reporting

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
</table>
### Privilege

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SELECT on schema _SYS_BIC</strong></td>
<td>This gives object-level access to activated data models. As long as all your data models require analytic privileges, it is safe to grant this. If you use data models which do not require analytic privileges, you should rather grant SELECT on individual activated models using the syntax in the next line. Note that in SAP HANA Live, only query views require Analytic Privileges, hence you must not grant SELECT on the entire _SYS_BIC schema when using SAP HANA Live.</td>
</tr>
<tr>
<td>catalog object &lt;package&gt;::&lt;object&gt;::SELECT</td>
<td>Gives SELECT on the activated version of the data model &lt;object&gt; in package &lt;package&gt;</td>
</tr>
<tr>
<td><strong>SELECT on &lt;conversion_tables&gt;</strong></td>
<td>If data models use currency conversion or unit of measure conversion, the end-users need SELECT access to the conversion tables.</td>
</tr>
<tr>
<td>Analytic privileges referring to the data models</td>
<td>Typically you can only read from an activated view if you have not only the SQL SELECT privilege, but also an analytic privilege referring to that view. If you are reading from a calculation view which again has other activated models as data sources, you also need analytic privileges referring to all these source models.</td>
</tr>
</tbody>
</table>

### 5.8.1.3 hdbrole file

The .hdbrole file looks as follows:

```bash
// role containing all generically required privileges
// for reporting against activated data models
// but not sufficient privileges to give any actual
// read access to business data
//
// users typically need the following privileges
// to report against data models:
//
// a) the "SELECT" privilege on the activated data model
//    (the object in _SYS_BIC)
// b) many BI clients require SELECT on metadata tables
//    in schema _SYS_BI
// c) potentially SELECT privilege on tables for
//    currency conversion or unit of measure conversion
// d) and a valid analytic privilege referencing the
//    data model.
//
// This role gives the only "safe" of the above privileges:
// SELECT on schema _SYS_BI
//
// If you plan to manage object-level and row-level access
// by means of analytic privileges, this role here can be
```

---

63
How To... Define Standard Roles for Administrators in SAP HANA Database

// granted to all reporting end-users, and you can then
// add user/group specific roles containing the appropriate
// analytic privileges.
// Privileges on currency and UOM conversion objects might
// be added here (if there is just one set of such objects)
// If there are multiple schemas with such objects (maybe
// for different application areas), you might add those
// privileges in the user-specific roles.
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.reporting::reporting_generic {
  // SQL SELECT access to all objects in schema _SYS_BI.
  // This is in fact only required for some model metadata
  // tables, but since _SYS_BI typically only contains
  // time dimension tables, there is normally no risk in
  // granting SELECT on the entire schema.
  catalog schema "_SYS_BI": SELECT;

  // SQL-level access to all activated data models
  // (remember: without analytic privilege, this does not
  // give data access)
  catalog schema "_SYS_BIC": SELECT;
  // Note: you might not like this privilege, and strictly,
  // you don't need it. You can also grant SELECT on
  // individual activated objects, preferably using the
  // syntax below. This is especially recommended if you have any
  // data models which do not require analytic privileges.
  catalog object <package>::object: SELECT;

  // schema for currency/unit of measure conversion:
  // If your data models include currency conversion or
  // UOM conversion, you have to give SELECT access to
  // the conversion tables.
  // Since we don't know where you place these tables, we
  // cannot include that privilege here.
  // we recommend including the privilege for the individual
  // tables (e.g. the TCUR* tables) only, instead of the
  // entire schema containing the tables. You never know
  // what other tables might be added into that schema later.
  // add the tables using the syntax (double quotes are required
  // and capitalization matters, wild cards are not allowed):
  catalog sql object "<schema_name>"."<table_name>": SELECT;

  // In addition to these object (SELECT) privileges, end-users
  // will require a valid analytic privilege referring to
  // the data model they want to read from.
  // include analytic privileges using the syntax:
  // analytic privilege: <package>;<privilege>.analyticprivilege;
}

5.8.2 Other end-user privilege requirements
What privileges your end users need depends on what they are allowed to do – and SAP HANA is an
extremely open system offering a wide variety of possibilities.
Next to reporting privileges, your end users might need some of the following:

### 5.8.2.1 Typical privileges required:

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application privilege</td>
<td>For starting XS applications or performing specific actions in these applications. Refer to the application documentation for details</td>
</tr>
<tr>
<td>Application-specific roles</td>
<td>Some applications come with end-user roles – again, see the application documentation</td>
</tr>
</tbody>
</table>

### 5.9 Roles for support

We foresee to groups of support roles:

- **Roles for basis support.**
  These are roles for support staff that is concerned with the database in SAP HANA, but not with the development platform (XS server, repository contents).
  Suggested roles are:
  - **ROLE BASIS_SUPPORT_READ**
  - **ROLE BASIS_SUPPORT_TRACE**

- **Roles for application support**
  These are roles intended for application support, i.e. people who will need to at least reading developer privileges to troubleshoot data models, XS applications, etc.
  Suggested roles are:
  - **ROLE APP_SUPPORT_READ**
  - **ROLE APP_SUPPORT_TRACE**
  - **ROLE APP_SUPPORT_READ_DATA**

Depending on the setup of your support organization, different support roles may be needed, e.g. including developer rights. In many cases, support staff may need a system administration or development role to fulfill their duties.

### 5.9.1 Role basis_support_read

Role for a read-only support user who can perform all typical DB monitoring tasks, such as monitoring memory and other resource consumption, system alerts, view traces, view system configuration, etc. but has no access to the repository.

This role is simply a reference to our basis role database_monitoring (5.3.7). We introduce this redundancy, so we can have dedicated name spaces for all roles to be granted to support staff. This will normally be a safe role to give to internal database support as well as to SAP support.

### 5.9.1.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege CATALOG READ</td>
<td>Read all metadata of the database catalog, including table metadata etc.</td>
</tr>
<tr>
<td>SELECT on schema_SYS_STATISTICS</td>
<td>View alerts from the statisticsserver</td>
</tr>
</tbody>
</table>
5.9.1.2 hdbrole file

The .hdbrole file looks as follows:

```
// Role that gives read access to all parts
// of a HANA DB that are required for typical
// DB monitoring (excluding the repository)
//
// The role presently only includes the admin role
// database_monitoring. The reason for the "redundancy"
// is that we want to have dedicated roles for support
// staff.
//
// Actions permitted by this role
// - Role database_monitoring
//   * Read access to all tabs of the administration
//     editor of SAP HANA studio
//   * Read access to the alerts of the statisticsserver
//   * Display the new graphical memory- and
//     resource-consumption displays

// unfortunately, we cannot include read access to
// the backup editor in studio
//
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.support.basis::basis_support_read
extends role sap-templates.security.common.roles.admin::database_monitoring
{
   -- no further privileges
}
```

5.9.2 Role basis_support_trace

Role for a read-only support user who can start/stop traces and perform all typical DB monitoring tasks, such as monitoring memory and other resource consumption, system alerts, view traces, view system configuration, etc. but has no access to the repository.

The role extends role basis_support_read and adds the TRACE ADMIN privilege.

Because of the trace management capabilities, we suggest that usage of this role should be restricted in more critical system landscapes, especially when working with external support such as SAP support staff.

5.9.2.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege TRACE ADMIN</td>
<td>Start/stop any database trace, change trace configuration of the database.</td>
</tr>
<tr>
<td></td>
<td>Side effect: also allows deleting trace files (files in tab diagnosis files)</td>
</tr>
</tbody>
</table>
How To... Define Standard Roles for Administrators in SAP HANA Database

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role sap-templates.common.roles.support.basis::basis_support_read</td>
<td>Overall database monitoring (except read access to repository)</td>
</tr>
</tbody>
</table>

5.9.2.2 hdbrole file

The hdbrole file looks as follows:

```
// Role that gives read access to all parts of a HANA DB that are required for typical DB monitoring (excluding the repository) and allows modifications of the database traces.

// The role presently only includes the admin role database_monitoring. The reason for the "redundancy" is that we want to have dedicated roles for support staff.

// Actions permitted by this role
- Role basis_support_trace
  - * enable start/stop of traces, trace configuration
  - + also allows deletion of trace files.
- Role database_monitoring (through basis_support_read)
  - * Read access to all tabs of the administration editor of SAP HANA studio
  - * Read access to the alerts of the statisticsserver
  - * Display the new graphical memory- and resource-consumption displays

// unfortunately, we cannot include read access to the backup editor in studio

// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.support.basis::basis_support_trace extends role sap-templates.security.common.roles.support.basis::basis_support_read {  
  -- switch on and off traces, change trace configuration, delete -- trace files  
  system privilege: TRACE ADMIN;  
}
```

5.9.3 Role app_support_read

Role for an application support user who can read all design-time information in the repository, i.e. all data model definitions, XS application source code, etc. but perform other actions. The user also has read access to the database catalog, so they can also check catalog object usage of these development artefacts. This will normally be a safe role to give to internal application support as well as to SAP support.

5.9.3.1 Granted Privileges

| Privilege | What does it do? |
5.9.3.2 **hdbrole file**

The .hdbrole file looks as follows:

```plaintext
// Read-only role for application support
// The role gives read access to the entire repository
// and to all metadata of the database.
// It does not include permission to execute any
// XS application or read from any data model etc.
// Actions permitted by this role
// - Role app_support_read
//   * access to the repository stored procedure
//   * read access to the entire repository
//   * read access to the catalog metadata
//   * access to the administration editor in
//     SAP HANA studio
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.support.app::app_support_read {
  -- allow access to the repository at all:
  catalog sql object "SYS"."REPOSITORY_REST": EXECUTE;
  -- read access to all packages in the repository:
  package ".REPO_PACKAGE_ROOT" : REPO.READ;
  -- read access to the metadata of the database
  -- catalog
  system privilege: CATALOG READ;
}
```

5.9.4 **Role app_support_trace**

Role for an application support user who can read all design-time information in the repository, i.e. all data model definitions, XS application source code, etc. but perform other actions.

The user also has read access to the database catalog, so they can also check catalog object usage of these development artefacts.

In addition, the user can start and stop traces, modify the trace configuration and view traces, as well as delete trace files.

Because of the trace management capabilities, we suggest that usage of this role should be restricted in more critical system landscapes, especially when working with external support such as SAP support staff.
5.9.4.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role sap-templates.security.common.support.app::app_support_read</td>
<td>Read all metadata of the database catalog, including table metadata etc.</td>
</tr>
<tr>
<td>System privilege TRACE ADMIN</td>
<td>Start/stop any database trace, change trace configuration of the database. Side effect: also allows deleting trace files (files in tab diagnosis files)</td>
</tr>
</tbody>
</table>

5.9.4.2 hdbrole file

The hdbrole file looks as follows:

```plaintext
// Read-only-role for application support including tracing capabilities
// The role gives read access to the entire repository and to all metadata of the database and which also allows tracing
// It does not include permission to execute any XS application or read from any data model etc.
// Actions permitted by this role
// - Role app_support_trace
//   * start/stop tracing, change trace configuration
//   * view traces
//   + also allows deletion of trace files
// - Role app_support_read
//   * access to the repository stored procedure
//   * read access to the entire repository
//   * read access to the catalog metadata
//   * access to the administration editor in SAP HANA studio
// Author: Richard Bremer <richard.bremer@sap.com>
role sap-templates.security.common.roles.support.app::app_support_trace
extends role sap-templates.security.common.roles.support.app::app_support_read
{
  -- switch on and off traces, change trace configuration, delete trace files
  system privilege: TRACE ADMIN;
}
```

5.9.5 Role app_support_read_data

Role for application support which not only gives read access to repository and database catalog, but also to the contents of all activated data models (unfiltered).
How To... Define Standard Roles for Administrators in SAP HANA Database

This role can be helpful, but in many cases unrestricted read access to application data will be going too far. We recommend using this role with care, if at all, in production instances (or instances containing productive data).

5.9.5.1 Granted Privileges

<table>
<thead>
<tr>
<th>Privilege</th>
<th>What does it do?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role sap-templates.security.common.support.app::app_support_read</td>
<td>Read all metadata of the database catalog, including table metadata etc.</td>
</tr>
<tr>
<td>Role sap-templates.security.common.development::model_preview_unrestricted</td>
<td>Full read access to all data models, including analytic privilege _SYS_BI_CP_ALL</td>
</tr>
</tbody>
</table>

5.9.5.2 hdbrole file

The .hdbrole file looks as follows:

```plaintext
// Read-only-role for application support, including data access
//
// The role gives read access to the entire repository
// and to all metadata of the database.
//
// And it includes full read access to all data models
// in the system. Note that this role must be strictly
// controlled in production systems, as it gives unfiltered
// access to data models.
//
// Actions permitted by this role
// - Role app_support_read_data
// - Role app_support_read
//   * access to the repository stored procedure
//   * read access to the entire repository
//   * read access to the catalog metadata
//   * access to the administration editor in
//     SAP HANA studio
// - Role model_preview_unrestricted
//   * full select and execute on schema _SYS_BIC
//   * full select on schema _SYS_BI
//   * the "see all" analytic privilege _SYS_BI_CP_ALL
//   * execute on the repository stored procedure REPOSITORY_REST
//
// Author: Richard Bremer <richard bremer@sap.com>
role sap-templates.security.common.roles.support.app::app_support_read_data
  extends role sap-templates.security.common.roles.support.app::app_support_read,
    sap-templates.security.common.roles.modeling::model_preview_unrestricted
  {
    -- no individual privileges added here
  }
```
5.9.6 Debugging
SAP HANA offers debugging capabilities for SQLScript procedures and SAP HANA XS Applications. For debugging, one needs user-specific privileges (to debug in another user’s session) and/or data specific privileges. It does not seem appropriate to attempt the definition of a generic debugging role.

5.9.7 Further support roles
In many cases you might find a need to reproduce issues a given named user in SAP HANA has. In these cases, it will often make sense to create a user copy of that named user and allow support staff to work with the copied user. As of Support Package Stack 7 (Revision 70 or higher) of SAP HANA, user copy is possible via functionality in SAP HANA Studio.

5.10 Role / System-tier matrix
Not all roles should be used (under normal circumstances) in all tiers of a system landscape. Let us assume a typical landscape with a development (DEV), test (Test), quality-assurance (QA) and production (PROD) system. The QA system typically is a more or less recent system copy from production, so it does typically contain sensitive data.

Any development task should be constrained to systems DEV and Test
The following table gives an overview of which roles should be (y), may be (m) and should not (x) be used in a given tier of the system landscape under normal circumstances. We also foresee some roles that shall only be used during installation/initial setup (xi)
We removed sap-templates.security from the package names for better readability.

<table>
<thead>
<tr>
<th>Package</th>
<th>Role</th>
<th>DEV</th>
<th>Test</th>
<th>QA</th>
<th>Prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected.roles</td>
<td>role_builder_native</td>
<td>y</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>role_editor_imported</td>
<td>m</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>role_builder</td>
<td>m</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>Security_developer_protected</td>
<td>m</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>user_admin</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>user_admin_unrestricted</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin_basic</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin_troubleshooting</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin_audit</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin_extended</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>security_admin_disk_encryption</td>
<td></td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>protected.roles</td>
<td>audit_operator</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>protected.roles</td>
<td>build_analytic_privilege_generic</td>
<td></td>
<td>y</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>basic_admin</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>persistence_admin</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>
How To... Define Standard Roles for Administrators in SAP HANA Database

<table>
<thead>
<tr>
<th>Package</th>
<th>Role</th>
<th>DEV</th>
<th>Test</th>
<th>QA</th>
<th>Prod</th>
</tr>
</thead>
<tbody>
<tr>
<td>common.roles.admin</td>
<td>backup_admin</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>system_admin_generic</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>system_admin_preinstalled</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>backup_operator</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>database_monitoring</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>repo_admin_production</td>
<td>m</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.admin</td>
<td>data_admin</td>
<td>m</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>repo_manager</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>repo_exporter</td>
<td>m</td>
<td>m</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>repo_importer</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>content_transport_manager</td>
<td>xi</td>
<td>xi</td>
<td>xi</td>
<td>xi</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>content_transport_executor</td>
<td>x</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.repository</td>
<td>content_transport_source</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>xs_app_dev_generic</td>
<td>y</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>xs_app_dev_public</td>
<td>y</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>data_modeling_generic</td>
<td>y</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>data_modeling_public</td>
<td>y</td>
<td>m</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>model_preview_unrestricted</td>
<td>y</td>
<td>y</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.development</td>
<td>build_analytic_privilege_public</td>
<td>y</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>protected.roles</td>
<td>build_analytic_privilege_generic</td>
<td>y</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>common.roles.support.basis</td>
<td>basis_support_read</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.support.basis</td>
<td>basis_support_trace</td>
<td>y</td>
<td>y</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>common.roles.support.app</td>
<td>app_support_read</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>common.roles.support.app</td>
<td>app_support_trace</td>
<td>y</td>
<td>y</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>common.roles.support.app</td>
<td>app_support_read_data</td>
<td>y</td>
<td>y</td>
<td>m</td>
<td>m</td>
</tr>
</tbody>
</table>
6. Appendix

The roles described in this document are available as a delivery-unit export, published on scn at the same location as the document itself.