# Saperion

## Update Guide

### Version: Foundation EP2

Written by: Documentation Team, R&D
Date: June 2020

Hyland™

## Copyright

# Table of Contents

# About the Saperion Foundation EP2 Update Guide

This guide provides information about updating Saperion from versions 7.5.x, 8.0.x, 8.1.x, and Foundation EP1 to Saperion Foundation EP2. Read this guide before you install the update.

Also read the Saperion version Foundation EP2 Technical Specifications, which list the complete system requirements and compatibilities, as well as discontinuations, for the current Saperion version.

We recommend completing the update procedure in a development environment before updating a production environment.

This document does not provide information about upgrades from versions prior to 7.5. If you need to upgrade from an earlier verison, contact Support.

## Important Note

Saperion version 8.0 and higher contain a new User Management, which requires that you perform certain tasks before you update your Saperion version. See the section “Migrate User Management” in this document for more information.

# Installation and update notes

## Update versions

We strongly recommend you have Saperion ECM version 7.5 SP6 or higher installed before you update to version Foundation EP2.

## .NET Framework

To install version Foundation EP2 of Saperion ECM, .NET Framework versions 3.5 and 4.x are required.

## Oracle

For information on Oracle, refer to the Setup topic under “Installation” on the documentation portal docs.hyland.com.

## Workflow Unicode support

Use the xdfdocvector_Unicode.ddc table.

## Mail applet JAR file

The Web Client applet for email has to be re-signed after an update, due to security settings in current Java versions. To receive a signed JAR file, send the following information to Support.

- Serial number
- Saperion version, including revision number, for example, 8.0.0.48836
- URLs used to call the Web Client (namely the URL to the Tomcat server)

### About client and server compatibility

When you update to Saperion version Foundation EP2 from a previous version, you must update the clients as well as the servers.

## Backups

Before beginning the update, back up the following.

- All server components
- Saperion installation directory

### Saperion Core Server

Also, complete the following procedures.

- Back up the database
- Back up the Broker DBS directory
- Back up all database definitions (*.DDC), forms (*.QBE), workflow definitions (*.DFD) and COLD definitions, as well as all Visual Basic scripts (*.BAS).
- Process all user trays
- Check all caches for open write entries and process them
- Back up media databases and the virtual media cache
- Check the disk space
- Make sure that original and mirror have identical states by processing all mirroring jobs.

### DDC files

The following Saperion system tables were changed since version 8.0. If you have customized these tables, back them up before the update and then restore them afterwards.

- Localize.ddc
- Pdfdocvector*.ddc
- xdfdocvector*.ddc
- xsuser_schema*.ddc
- xsgrp_schema*.ddc
- xsmnd_schema*.ddc

### Standard installation directory

If you are using a directory other than the standard Saperion installation directory for scenario files, such as definitions (DDC) or forms (QBE), note that the Saperion setup will ignore it. When you update, Saperion creates the following subdirectories and copies current system files there.

- defs (DDC files)
- macros (BAS files)
- forms (QBE files)
- workflow (DFD files)

## Web Client

### Update Version 7.5 to Foundation EP2

The ZK framework used by the Web Client was updated to the latest version, to add support for the latest browsers and web technologies. If you have customized themes, you must re-implement them for compatibility with the new framework. In addition, you may have to modify scripts used in forms in the Web Client to be compatible with API changes in the new framework.

## Java

### Java version 1.8 and 11

When using Java SDK 1.8, the following Oracle JDBC driver is required. Copy it to the /scr/scr-javacoreserver/lib directory.

- Ojdbc7.jar in the version 12.1.0.2.0

## Update Windows authentication

When you update from Saperion version 7.5.x to Foundation EP2, you must update the DLL file used for Windows authentication (sqljdbc_auth.dll). The file is included in the Microsoft JDBC SDK and can also be downloaded from the Microsoft website.

## Java Core Server

### Modules on the Java Core Server

The following modules are only available when using Java Core Server.

- Web Client
- ECM REST Services
- Integration Server (Business Rules Server, Step Performer)
- Applications based on the Classic Connector API
- File solutions

- Universal Importer

## Java Core Server in Saperion version Foundation EP2

In Saperion version 7.5, we released the Legacy Server, a 32-bit server that runs in a 64-bit environment. The Legacy Server takes over functions from the core server.

To use the following functions, you must install the 32-bit Legacy Server.

- Loading images that were stored in internal format (DocLoadOriginal=False) using the Classic Connector (Web Client)
- Broker EventScript tasks (a workaround via R/Link server is possible)
- Server-side workflow EventScript macros

## About transitioning to the Java Core Server

If the Saperion installation you are updating uses Java modules, you must transition the modules to the 64-bit Java Core Server. To do so, you must do a new install of the 64-bit Java Core Server and copy over the settings from the 32-bit system.

We describe the necessary procedures in this section. Where applicable, refer to the Saperion documentation for the detailed steps for each procedure. Or contact your representative for additional support with these tasks.

### Transition to the 64-bit system

This section describes how to transition a standard Saperion installation from a 32-bit to a 64-bit system.

### Before making the transition

Before you begin the transition, take the following precautionary steps.

- Back up your 32-bit Saperion installation, including all system and configuration files.
- Set up a 64-bit ODBC data source analogous to your existing 32-bit data source. The name of the new data source should correspond to the name of the 32-bit system (you can add this information later in the PROGRAM.INI file).
- Install a 64-bit Java 1.8 or 11 JDK. For more details, refer to the Saperion Technical Specifications.

### Install the 64-bit system

Refer to the Saperion documentation to guide you through the following installation procedures.

1. Install a complete Saperion 64-bit server.
2. Install the latest Saperion service pack or patch level.
3. Install the Saperion Legacy server, if needed.

### Configure the 64-bit system

To configure the 64-bit system, perform the following procedures.

1. Back up the archief.ini and program.ini files from your new 64-bit system.
2. From the 32-bit system backup, copy the archief.ini and program.ini files into the new 64-bit system.

3. From the 64-bit system PROGRAM.INI backup (step 1), copy the [modules64] section into the new 64-bit PROGRAM.INI file.
4. Copy all system files, such as DDC, QBE, DFD, and BAS, from the old system to the new system.
   **Note** This is optional for files with absolute paths.
5. From the 32-bit system backup, copy the [drive:]{path}\USERS\ directory, including the tray files as well as custom settings, to the new 64-bit system.
6. Make sure that the paths to the cache, media databases and repositories are all absolute paths or URLs. You may have to modify them to make them so.
7. Start the Java Core Server.
8. Start the Legacy server.
9. Using MMC, set up your broker macro tasks in the Legacy server.

### Test the 64-bit system

When your 64-bit system is up and running, make sure that all functions are working properly. In particular, check the following functionality.

1. The Web Client can access images that were stored without "DocLoadOriginal".
2. Workflows with server-side macros work as expected.

### FAQ 64-bit system

**I have broker macro tasks. If I install a 64-bit Java Core Server, can I still use the tasks?**

Yes, but you must install the Legacy32 Server.

**I have workflows with server-side macros. Can I use this workflow on a 64-bit Java Core Server?**

The Legacy Server can be used to execute workflows when server-side workflow scripts are contained in the workflow definition. Use the Legacy Server to minimize the work needed to transition to the 64-bit version of Saperion. We recommend migrating your scripts to Java. New script tasks should be created in Java and implemented in the Integration Server.

**Can I still start my old core server service?**

Because all new functions, such as ECM Services, Java Storage plug-ins, Classic Connector, or eFiles, are only available with the Java Core Server, the core server (without Java) is not supported anymore.

**Can I update my 32-bit Saperion core server to 64-bit?**

No, the 64-bit JCS needs a new installation, as described in the previous section.

# User Management

## Migrate User Management

From Saperion version 8.0, the User Management was overhauled. This section describes the various changes. To get started with the migration from version 7.5 to version Foundation EP2, see the section “Migrate from version 7.5 to Foundation EP2”.

## Changes overview

All existing applications continue to work with version Foundation EP2, without users having to modify existing custom formulas or implementations based on Saperion APIs. All data migration is automatically handled by Saperion when the system is first started. You may have to modify existing extensions to the user management, such as schema-extension DDCs and certain system settings, in your current 7.5 installation before you update to version Foundation EP2.

The old limits for ACLs and users listed below still apply to all 32-bit applications. This includes the COM and UBI APIs, as well as the Saperion Rich Client. If you plan to use 32-bit applications and the Web Client or the new Services mixed with the Classic Connector, see the “About 32-bit systems” section. At minimum, read the “Migrate from version 7.5 to Foundation EP2” section to understand all necessary prerequisites for a successful installation of version Foundation EP2.

## Restriction changes

Beginning with version 8.0, the following restrictions were removed.

- You can now apply ACLs to any number of groups or users. Previously, the limit was 200 per ACL.
- Users can be members of any number of groups. Previously, the limit was 150 groups per user.
- Two-step deletion has been implemented for all objects. You can no longer delete users, group or role objects that are in use. These remain in the database but are not displayed. For more information about deleted objects, see the “Deleting objects” topic in this document.

## Architecture changes

### Data model changes

With version 8.0, the User Management data model was overhauled.

The following tables were replaced or added.

| Object | Pre Version 8 | Version 8 |
|---|---|---|
| Users | XSUSR | XS_USERS |
| Groups | XSGRP | XS_GROUPS |
| ACLs | XSACL | XS_ACLS |
| Profiles / Roles | XSACC | XS_ROLES |
| Tenants | XSMND | XS_TENANTS |
| User in group references | (new) | XS_USER_GROUP |
| Group in group references | (new) | XS_GROUP_GROUP |
| ACL in group referencess | (new) | XS_ACL_GROUP |

| Object | Pre Version 8 | Version 8 |
|---|---|---|
| Links XS_ACL_GROUP with XS_GROUPS | (new) | XS_ACL_GROUP_RIGHTS |
| ACL in user references | (new) | XS_ACL_USER |
| Links XS_ACL_USER with XS_USERS | (new) | XS_ACL_USER_RIGHTS |

## Schema changes

The following new columns and uniqueness rules exist.

**ACLs (XSACL)**

- New column: ID whole number (Integer)
- New column: SysIndexState whole number (Integer)

**Roles (XSACC)**

- Changed column: AllowedCmdGroups (Whole number instead of String)

The table lists the bitmasks for role privileges.

| Privilege | Bitmask | Description |
|---|---|---|
| SSO | 0x400h | SSO is allowed |
| TenantAdmin | 0x200h | Admin for all tenants |
| Input | 0x100h | Data import in Rich Client allowed |
| Index | 0x080h | Create documents |
| Query | 0x040h | Search for documents |
| Show | 0x020h | Display documents |
| Define | 0x010h | Manage definitions (DDCs) |
| Config | 0x008h | Administer system |
| User | 0x004h | Use the User Management |
| Edit | 0x002h | Edit documents |
| OtherKoerbe | 0x001h | Access other trays in the Rich Client |

**Users (XSUSR)**

- Changed column: PwChanged (Timestamp instead of String)
- New unique column: DisplayName

**Groups (XSGRP)**

- New unique column: DisplayName

## Data integrity / Constraints

Beginning with version 8.0, Saperion uses database constraints in the User Management to ensure uniqueness in columns and in references between objects. This prevents data inconsistencies. Note that with data constraints, in some circumstances, objects cannot be deleted or modified. Also, you cannot create new objects that don't comply with uniqueness constraints.

Data integrity is achieved through the following unique columns.

| Object | Unique columns |
|---|---|
| Users | ID, ClientID+Fullname, ClientID+Displayname |
| Groups | ID, ClientID+Fullname |
| Roles | ID, ClientID+Name |
| ACL | ID, ClientID+Name |
| Client | ID, Name |

Referential integrity is ensured through the following references.

| Roles | Reference |
|---|---|
| User | used in ACL |
| Role | used in group |
| Group | used by user or another group |
| Client | uses user, group, ACL, or role |

**Note** Saperion does not test dynamic references (such as when a document is revised) of the ACL, user and group objects in index, workflow, log or lookup tables for referential integrity. Use the system configuration to secure these.

**About deleting objects**

Because operations fail when they contain references to deleted objects, all User Management objects are now implemented with a two-step deletion. When you delete an object, Saperion sets a marker in the

“deleted” column (previously SYSINDEXSTATE). The object remains in the system, but is not displayed. This protects the referential integrity.

To maintain integrity, never remove objects from the User Management. Instead, simply mark them as deleted. You can only permanently delete an object when it is not being used in a data object. For example, you cannot permanently delete a user who is still being used in an ACL.

You cannot use the API to permanently delete objects. You must carefully delete them in the database.

**Restore an object**

To restore a deleted object, complete the following steps.

1. In the client, navigate to **Administration** > **User** and select **Administration**.
2. In the **User Manager** dialog, select the corresponding tab for your object.
3. Select the **Deleted** checkbox.
4. Click **Result** to display all deleted objects.
5. Double-click the object you wish to restore.
6. Click **OK**.

## About 32-bit systems

The previous limits for ACLs and users familiar from previous versions of Saperion still apply to all 32-bit applications. This includes the COM and UBI APIs, as well as the Saperion Rich Client. This section describes what you should know if you plan to use 32-bit applications and the Web Client, or the new Services mixed with the Classic Connector.

### Compatibility views and triggers

Saperion version 8.0 and higher contain compatibility views so that the Rich Client and COM and UBI applications continue functioning as before. The views emulate the XSUSR, XSGRP, XSACC, XSACL and XSMND tables of the old User Management system. In addition, triggers are defined that guarantee a complete compatibility between the 32-bit applications and the new APIs Classic Connector/ECM Services.

Besides the compatibility views, the 32-bit applications use the new reference tables for users, groups and ACLs. For more information, see the “Automatic Migration” and the “Data Model” topics.

If you currently access these tables directly, make sure that you can also access the views directly. If you write directly to the tables without using the Saperion API, you must replace those functions with Saperion API calls.

### Objects outside of the 32-bit limitation

You can now create objects (user, group or ACL) in the User Management that go beyond the previous limitations of 200 rights per ACL and 150 groups per user or group. Such objects cannot always be used in 32-bit applications, when these applications are used in operations that make use of both 32-bit and 64-bit applications.

When an object cannot be used, the Java Core Server displays a warning when it starts, saying that the object cannot be used by 32-bit applications. For ACLs, this means that objects are not accessible. For users, it means that they cannot log in using 32-bit applications.

### 32-bit LDAP sync

In an LDAP sync, the system no longer deletes unavailable objects, but simply marks them as deleted, so that they do not affect the data integrity in a later sync.

## Programming changes

### Changes in system behavior

If you use the Saperion COM, UBI or Classic Connector APIs to fill the User Management, you must check your application to see whether it deletes objects and then creates an object with the same name later. Now, this operation will fail with a constraint violation error in the log. You must modify the implementation so that the system reuses the deleted object.

In addition, check whether the you are using the API to delete roles (previously profiles). If so, you must ensure that the roles are no longer referenced by groups. In version 8.0 and higher, the operation would fail with a constraint violation error in the log.

Also, duplicate display names are no longer allowed. Make sure that users always have unique display names. We recommend checking that the columns for your business logic are unique, for all objects. For more information, refer to the “Data Integrity / Constraints” topic.

# New implementations in 8.0

Use the ECM Services for all new implementations in projects wherever possible.

**Classic Connector**

The ECM Services replace all Classic Connector functions and provide a new Java API. Currently, some extensions, such as those in Web Client Events or in the Integration Server, may still be easier to implement when the events include the Classic Connector.

**UBI**

The UBI API is no longer supported and should not be used. It has been completely replaced with the RESTful ECM Services. An end-of-life date has not yet been set, so you do not have to modify existing applications at this time.

**COM**

COM applications continue to include limits for ACLs and users, so this API should not be used in new projects.

For more information about using COM API methods to search for deleted objects and set them to undeleted, see the iApplication methods SearchUser, SearchGroup, SearchACL and SearchRole in the COM documentation in the Saperion Developer documentation.

## About Object ID sequences

Saperion version 8.0 and higher uses the XS_USERMNGMNT_SEQ table to assign a new ID when you create a new User Management object. The table contains a row for each table of the User Management, in which the system maintains the next ID. Triggers ensure that ECM Services and 32-bit applications provide a correct, unique ID. Where it can, Saperion fills existing gaps in the object IDs.

### Supported databases

For information about which databases are supported, refer to the Saperion Technical Specifications document on the docs.hyland.com Documentation Portal.

# Micro Services

Saperion Foundation EP1 and higher include enhanced ECM Service implementation and introduces new REST based services. Each service is a Spring Boot application that can be run as a standalone Java application with java -jar <servicename>.war. Use the following ECM Services for all new implementations in projects wherever possible.

## Configuration Service

The Configuration Service is the central repository for configuration properties for the other services and must be set up first. The ZIP archive contains a config directory with YAML files. These files include default configuration properties for other services. By default, the Configuration Service uses a simple file system backend and serves the configuration properties from this directory. In a production environment, a centralized Git repository or a database should be used. This makes it possible to use more than one instance of the configuration service.

## Core Configuration Service

The Core Configuration Service is used by the ECM Service, Storage Service, and Authentication Service to load configuration data from Java Core Server and must be installed after Configuration Service is set up. This service must be installed on the system in which Java Core Server is installed.

## Authentication service

The Authentication Service loads its configuration properties from the Configuration Service. The default location of the Configuration Service is localhost:8888. When the Configuration Service is running on another system or port, or whenever the service is started as a standalone application, you must configure the URL in the file authenticationservice.xml by adding the command line argument -Dspring.cloud.config.uri=<http(s)://Host:port of config service>

## Storage Service

The Storage Service loads its configuration properties from the Configuration Service. The default location of the Configuration Service is localhost:8888. When the Configuration Service is running on another machine or port, you have to configure the URL in the file storageservice.xml by adding the command line argument -Dspring.cloud.config.uri=<http(s)://Host:port of config service>

## ECM Service

The ECM service, introduced in Saperion 8.0, is enhanced to support the new microservices architecture.

## ECMS Java Client SDK

### User Management

The user management resources and the corresponding Java clients now return more information when searching for the items. The search methods from the clients `GroupManagementServiceClient`,

`GroupMembershipClientForGroups`, `GroupMembershipClientForUsers`, `RoleManagementServiceClient`, `TenantManagementClient`, and `UserManagementServiceClient` returns the following new data types.

| Old | New |
|---|---|
| `Iterable<UserListItemType>` | `Iterable<UserType>` |
| `Iterable<GroupListItemType>` | `Iterable<GroupType>` |
| `Iterable<TenantListItemType>` | `Iterable<TenantType>` |
| `Iterable<RoleListItemType>` | `Iterable<RoleType>` |

**Note** For ACLs, the search method still returns `Iterable<AclListItemType>`.

### Tenant Management

In TenantManagement, you can update the tenant name along with the tenant description. Previously, you could only add tenant description as a parameter of the update() function.

### Tenant Client

In TenantClient, the following function is modified:

| Old | New |
|---|---|
| `void update(String description) throws EcmException;` | `void update(TenantPropertiesType properties) throws EcmException;` |

# Manage new features in Saperion Foundation EP2

## Render Service

Render Service helps you to convert any renderable file to PDF with OCR capabilities.The Render Service loads its configuration properties from the Configuration Service. The default location of the Render Service is localhost:8888. When the service is running on another machine or port, you have to configure the URL in the file renderservice.xml by adding the command line argument -Dspring.cloud.config.uri=<http(s)://Host:port of config service>

## HTTPS end-to-end

ECM services can be configured to communicate securely using the HTTPS protocol. Enabling this feature provide end-to-end security to all communications within the ECM services.

## Hyland Viewer

Hyland Viewer is a document viewer which displays documents using the File Conversion Service (a platform agnostic Restful API for file conversion).

### Virtual hosted addressing style

Saperion Foundation EP2 requires virtual-hosted style instead of the path style model as the addressing model to access objects using the S3 storage plugin. If you use path style model, Saperion automatically converts them to virtual-hosted style to minimize effort from your end.

### C# SDK

The C# SDK library is packed with improvements to enable users to create their own solution and integrate 3rd. party applications with the saperion platform. You can refer to sample code available in the ISO file at \products\saperion-client-cs-packaging-8.2.0-0\example directory.

### ShareBase integration

Integration of Saperion Foundation EP2 with Hyland ShareBase enables users to upload documents in Saperion directly to Hyland ShareBase.

## Migrate from previous versions to Foundation EP2

This section provides instruction on how to migrate from Saperion 7.5.6 and higher to Saperion Foundation EP2.

### Migrate from version 7.5

#### About automatic migration

When you first start the system after updating to version Foundation EP2, it automatically migrates the User Management data. All existing applications continue to work. You do not have to modify existing custom formulas or implementations based on Saperion APIs.

Make sure your system is prepared for the migration by first completing the procedures in the “Update from Version 7.5” section.

**Note** You must update the User Migration from Saperion version 7.5 SP6 or higher.

#### Important note

If the migration fails, the Java Core Server cannot start. Before you can continue, you must resolve any problems. Typical problems are data inconsistencies, or missing privileges for creating database objects.

As long as you have not created any new objects in the User Management, you can repeat the migration any number of times. All the data is backed up to *_BACKUP tables. If you make any changes before restarting a migration, you must apply the same changes to these tables. Before you restart the migration, delete all rows in the XS_MIGRATED table.

#### Prepare version 7.5 for migration

Before you update from version 7.5 to version Foundation EP2, you must complete these procedures, which are described in the following sections.

- Set User Management options
- Reset the user management encoding
- Convert schema extensions

- Ensure that S3 compatible cloud storage system supports Signature version 4
- Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model
- Run test scripts

## Set User Management options

You must store the User Management unencrypted to ODBC, to be able to use all functions of the Classic Connector and the Web Client. Make sure that this is the case before you update. For more information, see the “Encoding” description in “Setting up options for User Management” in the Saperion Administration User Management documentation.

## Reset the user management encoding

Before you update to version Foundation EP2, reset the encoding of your current user administration, by completing the following steps.

1. In a text editor, open the archief.ini file and locate the [System] section.
2. Set the AllowDecode parameter to TRUE.
3. Restart the core server.
4. In the Rich Client, navigate to **Administration** > **User** and click **Administration**.
5. In the **User Manager** dialog box, click the **Options** button.
6. In the **Settings** dialog box, make sure the **Encoded** check box is not selected. Click **OK**.
7. Exit Saperion and restart the server.
8. In the database table XSMND, remove the contents of the FKEY column.

## Convert schema extensions

Make sure that your existing schema extensions meet the following requirements for users, groups and tenants:

- All schema extension DDCs must contain the SYSROWID system field.
- The DDC name for user schema extension should be xusr_schema.ddc
- The DDC name for group schema extension should be xsgrp_schema.ddc
- The DDC name for tenant schema extension should be xsmnd_schema.ddc

**Note** When an ODBC connection is used, the tables xusr_schema.ddc, xsgrp_schema.ddc and sxmnd_schema.ddc must all be in the same database as the user administration tables. You can move the tables by changing the data source in the DDCs and then moving the data using SQL tools.

To check whether a DDC contains a specific system field, complete the following steps.

1. In the Rich Client, navigate to **Administration** > **User** and click **Administration**.
2. In the **User Manager** dialog box, click the **Options** button.
3. In the **Settings** dialog box, click the **Schema extension** tab, and double-click the desired table.
4. In the **Edit Tables Definition** dialog box, click the **Properties** tab and make sure that the SYSROWID system field check box is selected.
5. Click **OK** and exit Saperion.

6. To update the DDC in the database, call the table up in the database.

If the DDC names you use are different from those listed above, you must rename them and copy over the data.

## Ensure that S3 compatible cloud storage system supports Signature version 4

If you use Amazon S3 API request through S3 storage plugin, ensure that the S3 compatible systems support Signature version 4 signing process from AWS.

## Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model

If you access objects using the S3 storage plugin, ensure that your URL is in virtual-hosted style. If you use path style model, Saperion automatically converts them to virtual-hosted style to minimize effort from your end. To comply with the virtual-hosted style, your S3 bucket name must not contain the following components:

- Periods (.) - Virtual-hosted URLs are only supported for non-SSL (HTTP) requests. When you use URLs with SSL, the SSL wild-card certificate only matches buckets that do not contain periods.
- Non-routable names - Such names may include characters that are not valid as part of a domain name or may be case-sensitive.

**Note** You can overcome these issues by transferring data using the S3 batch operation feature.

## Run test scripts

Before you update, test your User Management to rule out that your tables contain data inconsistencies, such as duplicate names or IDs.

SQL test scripts for Oracle and MS SQL server are included in the “Data integrity test scripts” section of this document. The scripts check your data and log inconsistencies. The “Log inconsistencies” scripts also run as part of the automatic migration, but you can use them to resolve any issues before attempting the migration.

To run the test scripts, complete the following procedures.

1. Depending on which database you are using, copy the corresponding “Log inconsistencies” test script from the “Data integrity test scripts” section of this document to the clipboard.
2. Execute the script with a tool such as MS SQL Server Management Studio or Oracle SQL Developer. Run the script using the account with which Saperion also logs in to the database.
3. If the resulting list shown is not empty, resolve the found inconsistencies.

You can also find a SQL script to remove duplicates in the “Data integrity test scripts” section of this document. You must remove all other data inconsistencies manually from the User Management before updating.

# Migrate from version 8.0.x

## About automatic migration

When you first start the system after updating to version Foundation EP2, it automatically migrates the User Management data. All existing applications continue to work. You do not have to modify existing custom formulas or implementations based on Saperion APIs.

## Update to version Foundation EP2

After you have removed all inconsistencies, update to version Foundation EP2. Then, before starting the Java Core Server, complete these procedures, which are described in the following sections.

- Convert the User Management
- Create new JDBC data sources for User Management
- Configure Java built-in security to address deserialization vulnerability
- Ensure that S3 compatible cloud storage system supports Signature version 4
- Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model

## Convert the User Management

Before you start the Java Core Server for the first time after the update, complete the following steps.

1. Make sure that the account with which Saperion accesses the database has the following privileges.
    - CREATE/ALTER/DROP TABLE/VIEW/INDEX/STORED PROCEDURE/FUNCTION/TRIGGER
    - RENAME TABLE
    - INSERT INTO SELECT
2. Start the Java Core Server. The system converts your User Management and backs up your existing tables. All your applications can use the new User Management without further configuration.
3. If the service does not start, see the “Important note” in the “About automatic migration” topic for further information and approaches.

## Create new JDBC data sources for User Management

The first update will fail if you have kept the User Management on a separate ODBC source without defining a corresponding JDBC data source in the archief.ini file. For technical reasons, we recommend working with one data source for all tables. For security reasons, however, it may be necessary to store User Management tables to a separate account. In this case, you need to add an additional data source to the dataSourcesConfiguration.xml file, so that the Java Core Server can find it.

To execute the update successfully after the initial fail, complete the following procedures.

1. In a text editor, open the [drive:]{path}scr\scr-javacoreserver\config\dataSourcesConfiguration.xml file, and enter a JDBC data source. For more information about data source configuration, see the “About Java Core Server options” topic in the Saperion Installation guide.
2. Restart the server.
3. Optional. Create or modify an ODBC data source.

4. In the Rich Client, open the **User Manager** dialog box and click **Options** to configure the data source.
5. When you are done, restart the server.

## Configure Java built-in security to address deserialization vulnerability

To ensure that Java built-in security is enabled to address the deserialization vulnerability, complete the following steps.

1. Remove the following settings in the saperion.properties file.
   - blacklisted.classes.for.deserialization - Includes a list of blacklisted classes. If Saperion detects deserialization request from any of the listed classes, it generates an exception. A sensible default has been provided.
   - max.allowed.array.size - Specifies the maximum size of an array or array list that can be deserialized and blocks a larger request by generating an exception.
   - max.allowed.map.size - Specifies the maximum entries of a map implementation that is allowed for deserialization and blocks a larger request with an exception.
2. Activate and list the serialization filters in the Saperion Java Core Server. To activate and list the serialization filters in the Saperion Java Core Server, complete the following steps.
   - In the [drive:]\\{path}\Saperion\Application\scr\scr-javacoreserver\config directory, open the "wrapper.conf" file.
   - Under the Java Additional Parameters section, add the `-Djdk.serialFilter` parameter.
   - Example:
   - # Java Additional Parameters
   - wrapper.java.additional.6=Djdk.serialFilter=maxarray=1000000;maxdepth=1000;maxrefs=1000000;!org.apache.commons.collections.functors.InvokerTransformer;!com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;!org.apache.commons.collections.functors.InstantiateTransformer;!org.codehaus.groovy.runtime.ConvertedClosure;!org.codehaus.groovy.runtime.MethodClosure;!org.apache.log4j.net.SocketServer

## Ensure that S3 compatible cloud storage system supports Signature version 4

If you use Amazon S3 API request through S3 storage plugin, ensure that the S3 compatible systems support Signature version 4 signing process from AWS.

## Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model

If you access objects using the S3 storage plugin, ensure that your URL is in virtual-hosted style. If you use path style model, Saperion automatically converts them to virtual-hosted style to minimize effort from your end. To comply with the virtual-hosted style, your S3 bucket name must not contain the following components:

- Periods (.) - Virtual-hosted URLs are only supported for non-SSL (HTTP) requests. When you use URLs with SSL, the SSL wild-card certificate only matches buckets that do not contain periods.

- Non-routable names - Such names may include characters that are not valid as part of a domain name or may be case-sensitive.

**Note** You can overcome these issues by transferring data using the S3 batch operation feature.

# Migrate from version Foundation EP1

## About automatic migration

When you first start the system after updating to version Foundation EP2, it automatically migrates the User Management data. All existing applications continue to work. You do not have to modify existing custom formulas or implementations based on Saperion APIs.

## Update to version Foundation EP2

After you have removed all inconsistencies, update to version Foundation EP2. Then, before starting the Java Core Server, complete these procedures, which are described in the following sections.

- Configure Java built-in security to address deserialization vulnerability
- Ensure that S3 compatible cloud storage system supports Signature version 4
- Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model

## Configure Java built-in security to address deserialization vulnerability

To ensure that Java built-in security is enabled to address the deserialization vulnerability, complete the following steps.

1. Remove the following settings in the saperion.properties file.
    - blacklisted.classes.for.deserialization - Includes a list of blacklisted classes. If Saperion detects deserialization request from any of the listed classes, it generates an exception. A sensible default has been provided.
    - max.allowed.array.size - Specifies the maximum size of an array or array list that can be deserialized and blocks a larger request by generating an exception.
    - max.allowed.map.size - Specifies the maximum entries of a map implementation that is allowed for deserialization and blocks a larger request with an exception.
2. Activate and list the serialization filters in the Saperion Java Core Server. To activate and list the serialization filters in the Saperion Java Core Server, complete the following steps.
    1. In the [drive:]\\{path}\Saperion\Application\scr\scr-javacoreserver\config directory, open the "wrapper.conf" file.
    2. Under the Java Additional Parameters section, add the `-Djdk.serialFilter` parameter.
    - **Example:**
    - `# Java Additional Parameters`
    - `wrapper.java.additional.6=Djdk.serialFilter=maxarray=1000000;maxdepth=1000;maxrefs=1000000;!org.apache.commons.collections.functors.InvokerTransformer ;!com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;!org.apache.commons.collections.functors.InstantiateTransformer`

```
;!org.codehaus.groovy.runtime.ConvertedClosure;!org.codehaus.groovy.run
time.MethodClosure;!org.apache.log4j.net.SocketServer
```

- 

### Ensure that S3 compatible cloud storage system supports Signature version 4

If you use Amazon S3 API request through S3 storage plugin, ensure that the S3 compatible systems support Signature version 4 signing process from AWS.

### Ensure that the S3 bucket name complies with the bucket naming policy of virtual hosted model

If you access objects using the S3 storage plugin, ensure that your URL is in virtual-hosted style. If you use path style model, Saperion automatically converts them to virtual-hosted style to minimize effort from your end. To comply with the virtual-hosted style, your S3 bucket name must not contain the following components:

- Periods (.) - Virtual-hosted URLs are only supported for non-SSL (HTTP) requests. When you use URLs with SSL, the SSL wild-card certificate only matches buckets that do not contain periods.
- Non-routable names - Such names may include characters that are not valid as part of a domain name or may be case-sensitive.

**Note** You can overcome these issues by transferring data using the S3 batch operation feature.

## Automatic migration

This section lists the steps that the system executes when you start the Java Core Server for the first time after updating to version Foundation EP2. If there is an error in any of the migration actions – migrating clients, roles, groups, users, ACLs – the system stops the migration and the server start fails. Make sure you have completed all the steps in the “Prepare” and “Update” topics.

### Migration description

When you start the Java Core Server for the first time, Saperion completes the following procedures.

1. Creates the new XS_* tables, if they do not yet exist. For more information, see the “Data Model” topic at the end of this document.
2. Creates necessary TRIGGER, FUNCTIONS und STORED PROCEDURES.
3. Deletes existing views that ensure compatibility with 32-bit applications (XSUSR, XSGRP, XSACC, XSACL, XSMND).
4. Renames existing backup tables XS_USR_BACKUP, XSGRP_BACKUP, XSACC_BACKUP, XSACL_BACKUP and XSMND_BACKUP to XSUSR, XSGRP, XSACC, XSACL and XSMND respectively.
5. Creates the migration tables XS_MIGRATED and XS_MIGRATION_CHECK, if these do not exist.
6. Checks whether the migration was already executed successfully by reviewing the XS_MIGRATED table. If the migration was successful, the system creates the compatibility views.
7. Checks the data consistency. Logs inconsistencies in the XS_MIGRATION_CHECK table. The system empties the table before the check, and stops the check when inconsistencies are found.

8. If necessary, extends the XSACL table with the new ID and SYSINDEXSTATE fields, and initializes the columns with a consecutive number beginning with 1, and the SYSINDEXSTATE with 0 (not deleted).
9. Migrates clients.
    3. Creates a default client with ID 0, which is marked as deleted.
    4. Copies all clients from XSMND to XS_TENANTS
    5. Writes the number of converted clients in the XS_MIGRATED table (from the XS_TENANTS table).
    6. Creates a backup, and renames XSMND to XSMND_BACKUP.
10. Migrates roles (previously called profiles)
    7. Creates a default role with ID 0, and marks it as deleted.
    8. Copies all roles from XSACC to XS_ROLES.
    9. Writes the number of converted roles in the XS_MIGRATED table (from the XS_ROLES table)
    10. Creates a backup and rename XSACC to XSACC_BACKUP.
11. Migrates groups
    11. Creates a default group with ID 0, and marks it as deleted.
    12. Copies all groups from XSGRP to XS_GROUPS.
    13. Copies references to other groups to the XS_GROUP_GROUP table. The system ignores references to groups that no longer exist.
    14. Writes the number of converted groups in the XS_MIGRATED table (from the XS_GROUPS table).
    15. Writes the number of converted group references in the XS_MIGRATED table (from the GROUP_GROUP table).
    16. Creates a backup and renames XSGRP to XSGRP_BACKUP.
12. Migrates users
    17. Creates a default group with ID 0, and marks it as deleted.
    18. Copies all users from XSUSR to XS_USERS.
    19. Copies the group references in the XS_USER_GROUP table. The system ignores references to groups that no longer exist.
    20. Writes the number of converted users in the XS_MIGRATED table (from the XS_USERS table).
    21. Writes the number of converted group references in the XS_MIGRATED table (from the XS_USER_GROUP table).
    22. Creates a backup and renames XSUSR to XSUSR_BACKUP.
13. Migrates ACLs
    23. Creates a default ACL with ID 0, and marks it as deleted.
    24. Copies all ACLs from XSACL to XS_ACLS. Does not migrate ACLs with invalid clients.

25. Copies the group permissions in the XS_ACL_GROUP and XS_ACL_GROUP_RIGHTS tables. The system ignores references to groups that no longer exist.
26. Writes the number of converted ACLs in the XS_MIGRATED table (from the XS_ACLS table).
27. Creates a backup and renames XSACL to XSACL_BACKUP.

14. Updates the admin role with the privilege to administer clients. This privilege is new and is evaluated by the ECM Services.
15. Creates the compatibility views XSUSER, XSGRP, XSACC, XSACL and XSMND.

## Data integrity test scripts

Before you update, test your current User Management data integrity by running one of the following “Log inconsistencies” scripts.

To run a test script, complete the following steps.

1. Copy the appropriate script to the clipboard.
2. Use a tool such as MS SQL Server Management Studio or Oracle SQL Developer to execute the script. You must use the account with which Saperion also logs in to the database.

### Scripts for MS SQL Server

Use these MS SQL scripts to log inconsistencies or repair duplicate names due to deleted records.

#### Log inconsistencies

```
IF OBJECT_ID('XS_MIGRATION_CHECK', 'U') IS NULL CREATE TABLE [XS_MIGRATION_CHECK] (
[NAME] varchar(128), [TENANT] int, [ERROR] varchar(128) )

GO

DELETE [XS_MIGRATION_CHECK]

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [FULLNAME], [SYSCLIENT],
'DUPLICATE FULLNAME IN XSUSR' FROM [XSUSR] GROUP BY [FULLNAME], [SYSCLIENT] HAVING
COUNT(*) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [ID], MAX([SYSCLIENT]),
'DUPLICATE ID IN XSUSR' FROM [XSUSR] GROUP BY [ID] HAVING COUNT([ID]) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [FULLNAME], [SYSCLIENT],
'DUPLICATE FULLNAME IN XSGRP' FROM [XSGRP] GROUP BY [FULLNAME], [SYSCLIENT] HAVING
COUNT(*) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [ID], MAX([SYSCLIENT]),
'DUPLICATE ID IN XSGRP' FROM [XSGRP] GROUP BY [ID] HAVING COUNT([ID]) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [NAME], [SYSCLIENT],
'DUPLICATE FULLNAME IN XSACC' FROM [XSACC] GROUP BY [NAME], [SYSCLIENT] HAVING
COUNT(*) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [ID], MAX([SYSCLIENT]),
'DUPLICATE ID IN XSACC' FROM [XSACC] GROUP BY [ID] HAVING COUNT([ID]) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [ACLNAME], [SYSCLIENT],
'DUPLICATE FULLNAME IN XSACL' FROM [XSACL] GROUP BY [ACLNAME], [SYSCLIENT] HAVING
COUNT(*) > 1
```

```
INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [NAME], 0, 'DUPLICATE
FULLNAME IN XSMND' FROM [XSMND] GROUP BY [NAME] HAVING COUNT(*) > 1

INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT [ID], 0, 'DUPLICATE ID
IN XSMND' FROM [XSMND] GROUP BY [ID] HAVING COUNT(*) > 1

SELECT * FROM [XS_MIGRATION_CHECK]
```

### Repair duplicate names due to deleted records; add “Deleted: <timestamp>”

```
UPDATE [XSGRP] SET [FULLNAME] = [FULLNAME] +  ', deleted: ' +
CONVERT(varchar,[SYSTIMESTAMP]) WHERE [SYSINDEXSTATE] > 0  AND [FULLNAME] IN (SELECT
[FULLNAME] FROM [XSGRP] GROUP BY [FULLNAME], [SYSCLIENT] HAVING COUNT(*) > 1 )

UPDATE [XSUSR] SET [DISPLAYNAME] = [DISPLAYNAME] +  ', deleted: ' +
CONVERT(varchar,[SYSTIMESTAMP]) WHERE [SYSINDEXSTATE] > 0  AND [DISPLAYNAME] IN
(SELECT [DISPLAYNAME] FROM [XSUSR] GROUP BY [DISPLAYNAME], [SYSCLIENT] HAVING COUNT(*)
> 1 )

UPDATE [XSUSR] SET [FULLNAME] = [FULLNAME] +  ', deleted: ' +
CONVERT(varchar,[SYSTIMESTAMP]) WHERE [SYSINDEXSTATE] > 0  AND [FULLNAME] IN (SELECT
[FULLNAME] FROM [XSUSR] GROUP BY [FULLNAME], [SYSCLIENT] HAVING COUNT(*) > 1 )
```

## Scripts for Oracle

Use these Oracle scripts to log inconsistencies or repair duplicate names due to deleted records.

### Log inconsistencies

```
BEGIN

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT
"FULLNAME", "SYSCLIENT", ''DUPLICATE FULLNAME IN XSUSR'' FROM XSUSR GROUP BY
"FULLNAME", "SYSCLIENT" HAVING COUNT(*) > 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT "ID",
MAX("SYSCLIENT"),''DUPLICATE ID IN XSUSR'' FROM XSUSR GROUP BY "ID" HAVING COUNT("ID")
> 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT
"FULLNAME","SYSCLIENT", ''DUPLICATE FULLNAME IN XSGRP'' FROM XSGRP GROUP BY
"FULLNAME", "SYSCLIENT" HAVING COUNT(*) > 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT "ID",
MAX("SYSCLIENT"), ''DUPLICATE ID IN XSGRP'' FROM XSGRP GROUP BY "ID" HAVING
COUNT("ID") > 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT "ID",
MAX("SYSCLIENT"),''DUPLICATE ID IN XSACC'' FROM XSACC GROUP BY "ID" HAVING COUNT("ID")
> 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT
"NAME", "SYSCLIENT",''DUPLICATE NAME IN XSACC'' FROM XSACC GROUP BY ("NAME",
"SYSCLIENT") HAVING COUNT(*) > 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT
"ACLNAME", MAX("SYSCLIENT"),''DUPLICATE ID IN XSACL'' FROM XSACL GROUP BY "ACLNAME"
HAVING COUNT(*) > 1';

EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT
"NAME", 0,''DUPLICATE NAME IN XSMND''  FROM XSMND GROUP BY "NAME" HAVING COUNT("NAME")
> 1';
```

```
EXECUTE IMMEDIATE 'INSERT INTO XS_MIGRATION_CHECK ( NAME, TENANT, ERROR ) SELECT "ID",
0,''DUPLICATE ID IN XSMND'' FROM XSMND GROUP BY "ID" HAVING COUNT("ID") > 1';

EXCEPTION

           WHEN OTHERS THEN NULL;

END
```

### Show migration integrity problems

```
SELECT * FROM XS_MIGRATION_CHECK
```

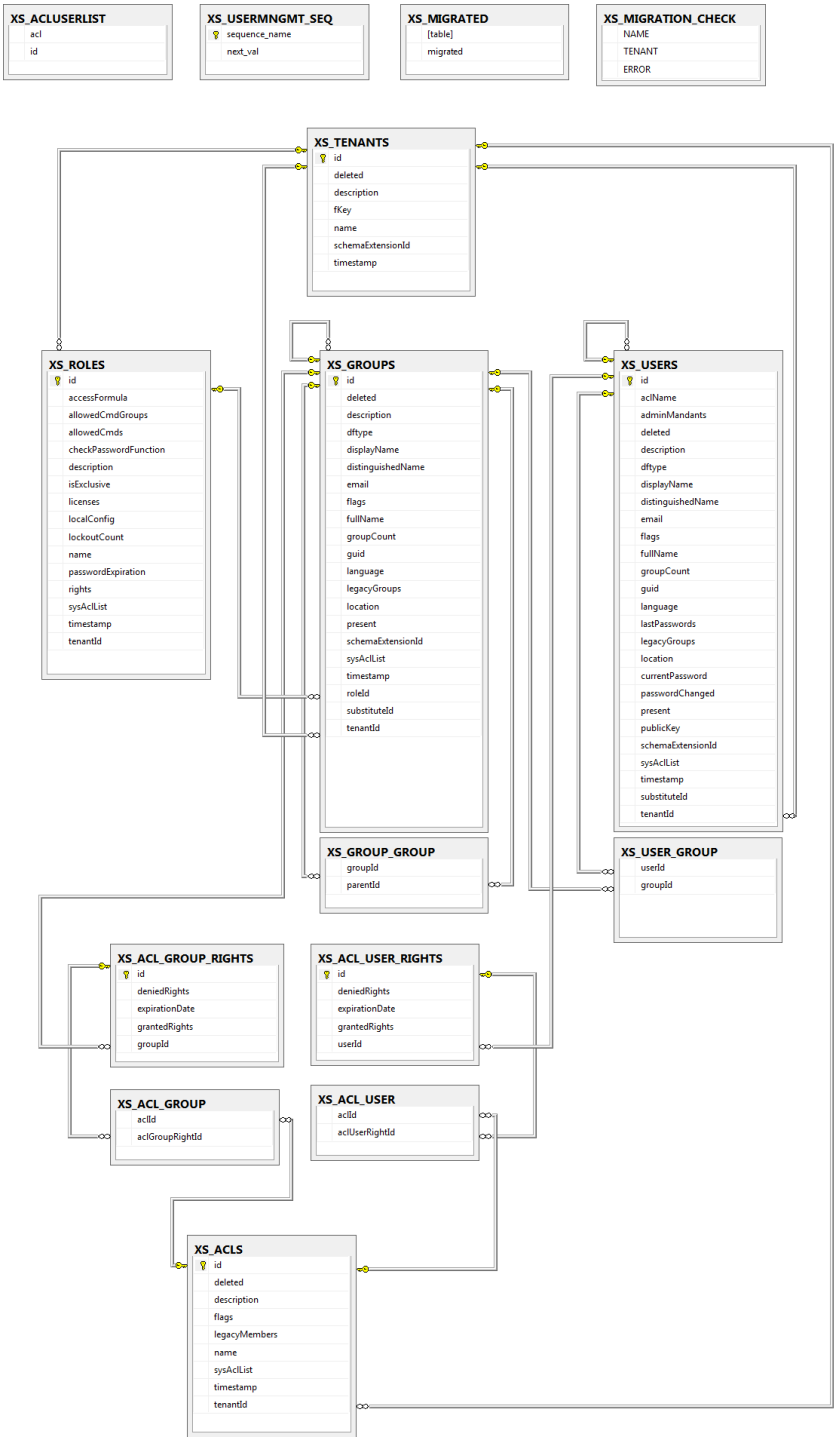
### Repair duplicate names due to deleted records; add “Deleted: <timestamp>”

```
UPDATE XSGRP SET FULLNAME = FULLNAME ||  ', deleted: ' || TO_CHAR( SYSTIMESTAMP) WHERE
SYSINDEXSTATE > 0  AND FULLNAME IN (SELECT FULLNAME FROM XSGRP GROUP BY FULLNAME,
SYSCLIENT HAVING COUNT(*) > 1 )

UPDATE XSUSR SET FULLNAME = FULLNAME ||  ', deleted: ' || TO_CHAR( SYSTIMESTAMP) WHERE
SYSINDEXSTATE > 0  AND FULLNAME IN (SELECT FULLNAME FROM XSUSR GROUP BY FULLNAME,
SYSCLIENT HAVING COUNT(*) > 1 )
```

## Data Model

The following Entity Relationship Diagram (ERD) shows the relationships between the standard tables used in the new User Management.


XS_ACLUSERLIST
acl
id
XS_USERMNGMT_SEQ
sequence_name
next_val
XS_MIGRATED
[table]
migrated
XS_MIGRATION_CHECK
NAME
TENANT
ERROR
XS_TENANTS
id
deleted
description
fKey
name
schemaExtensionId
timestamp
XS_ROLES
id
accessFormula
allowedCmdGroups
allowedCmds
checkPasswordFunction
description
isExclusive
licenses
localConfig
lockoutCount
name
passwordExpiration
rights
sysAclList
timestamp
tenantId
XS_GROUPS
id
deleted
description
dftype
displayName
distinguishedName
email
flags
fullName
groupCount
guid
language
legacyGroups
location
present
schemaExtensionId
sysAclList
timestamp
roleId
substituteId
tenantId
XS_USERS
id
aclName
adminMandants
deleted
description
dftype
displayName
distinguishedName
email
flags
fullName
groupCount
guid
language
lastPasswords
legacyGroups
location
currentPassword
passwordChanged
present
publicKey
schemaExtensionId
sysAclList
timestamp
substituteId
tenantId
XS_GROUP_GROUP
groupId
parentId
XS_USER_GROUP
userId
groupId
XS_ACL_GROUP_RIGHTS
id
deniedRights
expirationDate
grantedRights
groupId
XS_ACL_USER_RIGHTS
id
deniedRights
expirationDate
grantedRights
userId
XS_ACL_GROUP
aclId
aclGroupRightId
XS_ACL_USER
aclId
aclUserRightId
XS_ACLS
id
deleted
description
flags
legacyMembers
name
sysAclList
timestamp
tenantId