Administration Guide | PUBLIC
SAP IQ 16.1 SP 05
Document Version: 1.0.0 – 2026-01-08

SAP IQ Administration: User Management and Security

Content

1 SAP IQ Administration: User Management and Security

User and security implementation and administration, and external authentication methods.

2 Plan and Implement Role-Based Security

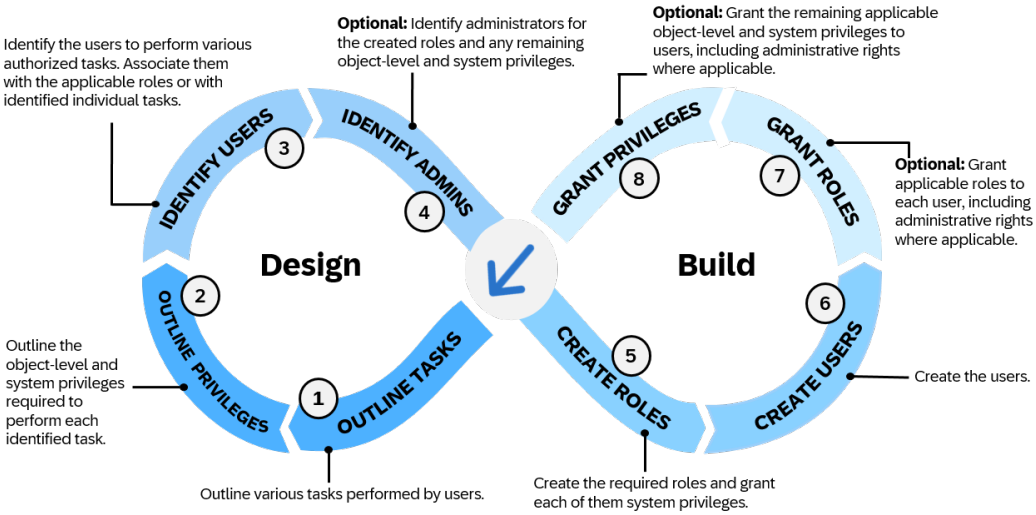
There is a distinct workflow to planning and implementing a role-based security model.

Designing the Security Hierarchy

1. Identify the various authorized tasks to be performed by users. Group closely related tasks. Groupings can be based on any organizational structure—departmental, functional, and so on. You can create a role hierarchy that matches the organizational hierarchy. Assign a name to each grouping. These groupings represent the roles you create.
2. Identify the system privileges and object-level privileges required to perform each authorized task identified.
3. Identify the users to perform the various authorized tasks. Associate them with the applicable roles or with identified individual tasks.
4. (Optional) Identify administers for the roles you create. Administrators can grant and revoke the role to other users.
5. (Optional) Identify administrators for the system privileges and object-level privileges that are not part of the roles you create.

Build the Security Hierarchy

1. Create the required roles. See *Roles*.
2. To each role, grant the system privileges. See *Roles* and *Privileges*.
3. Create the users. See *Users*.
4. Grant applicable roles to each user, including administrative rights where applicable. See *Roles*.
5. Grant applicable object-level and system privileges to users (not already indirectly granted through roles), including administrative rights where applicable. See *Privileges*.



- Privileges [page 48]
- Roles [page 87]
- Users [page 11]
- Managing Roles [page 116]
- Managing Roles [page 116]
- Users [page 11]
- Privileges [page 48]
- Roles [page 87]

Related Information

Roles [page 87]
Privileges [page 48]
Users [page 11]

3 User Management

SAP IQ provides a role-based security model for controlling access to database objects and executing privileged operations. This model provides complete control and granularity for the privileges you want to grant to users. Each privileged operation in a database requires one or more system or object-level privileges be assigned to the user to execute the operation.

A **system privilege** allows users to perform authorized database tasks. For example, assign the CREATE TABLE system privilege to a user to allow him or her to create self-owned tables.

An **object-level privilege** allows a user to perform an authorized task on a specified object. For example, assign ALTER object-level privilege on `TableA` to a user to allow him or her to alter that table, but no other tables.

A **role** is a container that may contain one or more system privileges, object-level privileges, and other roles. Granting a role to a user is equivalent to granting the user the underlying system and object-level privileges of the role.

All new users are automatically granted the PUBLIC system role, which gives them the ability to:

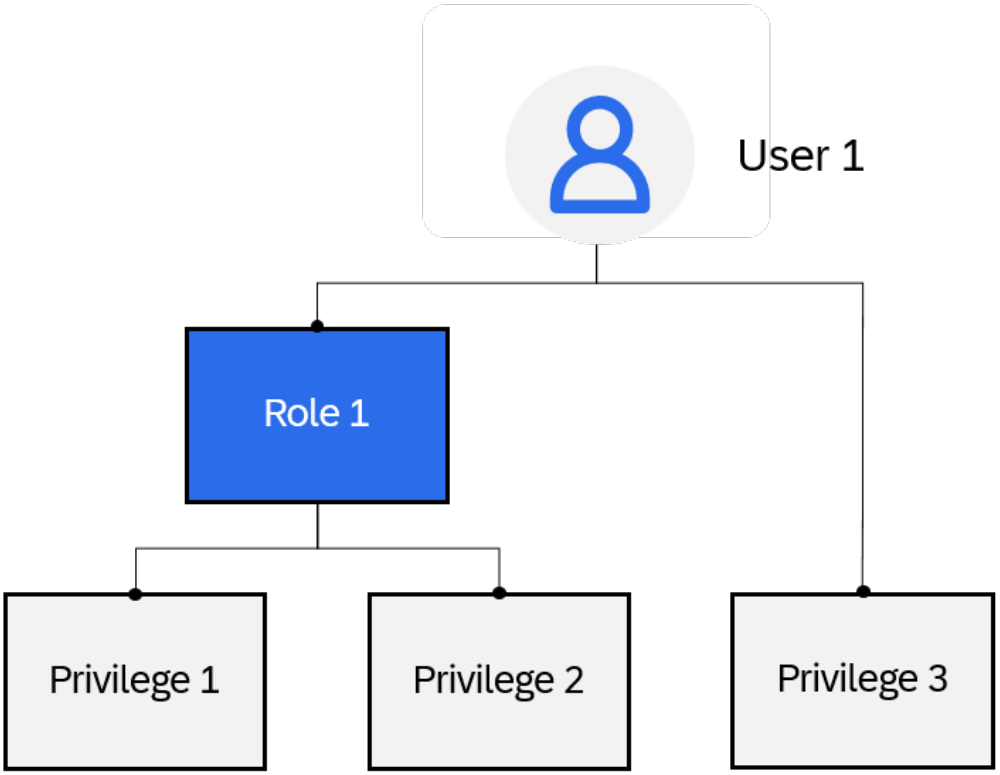
- View the data stored in the system views
- Execute most system stored procedures

Once you have created a new user, you can:

- Grant user-defined roles, system roles, system privileges, and object-level privileges to it.
- Assign a login policy it. By default, a user is assigned to the root login policy.
- Set it as the publisher or as a remote user of the database for use in a SQL Remote system.

Each new or migrated SAP IQ database includes a predefined set of roles you can use to get started. These system roles act as a starting point for implementing role-based security.

> **Note**
>
> If you have used versions of SAP IQ earlier than 16.0, you should review *Role-Based Security Model* in the *SAP IQ Installation and Update Guide* for your operating system, which describes how the security model has changed from using authorities, permissions, object-level permissions, and groups to a role-based security model that uses roles, system privileges, object-level privileges, and user-extended roles.



In this section:

3.1 Users

User management includes the creation and deletion of user IDs, as well as password management.

User Defined

User defined accounts are used for general access to system functionality and services.

- Created manually by a user or admin.
- Require a login and a user-managed password.
- Can be deleted by authorized users or administrators.
- Suitable for regular users.

System Created

The DBA user is a high-privilege administrative user. It is the default user.

- Created automatically by the system.
- Has login capability and a system-assigned or admin-managed password.
- Can be deleted by an administrator.
- Intended for critical system administration and setup tasks.
- Serves as the primary user with administrative rights in the system.

In this section:

Remove a user ID from the database.

Related Information

Plan and Implement Role-Based Security [page 7]
Modifying an Existing Login Policy [page 141]

3.1.1 DBA (Default) User

The DBA user, is the default user created when a new SAP IQ database is created.

Sometimes referred to as the superuser, DBA is a highly privileged user who can exercise any system privilege, administer any role, and perform any privileged operation in the database. For this reason, DBA should not be used on a daily basis to administer the database. The user name and initial password are specified during database creation. By default, the password never expires; therefore, SAP recommends changing the DBA password on a regular basis.

The DBA user is automatically granted administrative rights on the SYS_AUTH_DBA_ROLE role, which in turn is granted the roles SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE. The union of these roles grants the DBA user all system and object-level privileges, enabling full administrative control over the database.

To allow the DBA user to act as a superuser, all new user-extended and user-defined roles must be granted to the DBA user with administrative rights. Administrative rights can be granted as either a role administrator or a global role administrator.

The DBA user is widely known as the default administrative user. This makes it a prime target for unauthorized access attempts.To ensure database security and accountability, SAP recommends that you create a new administrative user, grant that user all the rights and privileges of DBA (see Create a DBA Equivalent User [page 22]), and then lock the DBA user (see Locking the DBA User [page 23]).

The SYS_AUTH_DBA_ROLE Role

Under certain circumstances, the underlying roles of SYS_AUTH_DBA_ROLE role can be dropped, and the underlying system privileges of the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE roles revoked. This can result in the DBA user being unable to perform specific privileged operations. For this reason, SAP strongly recommends that all underlying roles and system privileges of the SYS_AUTH_DBA_ROLE role remain as granted by default.

To guard against password loss by the active DBA user, create one or more equivalent accounts (with a uniquely generated user name and password) and lock up those credentials. If the active DBA password is lost, use one of the extra credentials to log in to that DBA account, and reset the original account password.

Protecting Against Password Loss

To reduce the risk of password loss for active users, grant password-change authority to only a small number of administrators. Users do not need any privilege to change their own passwords.

Grant the `CHANGE PASSWORD` system privilege only as broadly as necessary: for specific users or roles, or for all users or roles. Grant it with the `WITH ADMIN OPTION` only if recipients must delegate the privilege. Also assign these users a login policy with the `change_password_dual_control` option enabled.

The `CHANGE PASSWORD` system privilege is required to change the passwords of DBA and any DBA-equivalent user. Limit this privilege for these administrative accounts to a very small group, ideally two to four users.

If you create DBA-equivalent users, store their passwords securely and rotate them regularly. Do not use these administrative accounts for routine or daily tasks.

In this section:

Related Information

Alphabetical List of System Privileges [page 50]
Alphabetical Listing of Object Privileges [page 73]
Login Policies [page 136]
GRANT CHANGE PASSWORD Privilege Statement

3.1.1.1 System Privileges Granted to DBA

A list of all system privileges granted to the DBA user in SAP IQ.

All of the DBA user's privileges are granted through membership in the SYS_AUTH_DBA_ROLE system role. The SYS_AUTH_DBA_ROLE is comprised of the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE system roles. It is the combination of these two roles that give the DBA user all possible privileges in the database.

SYS_AUTH_SA_ROLE role contains the following privileges:

All grants are with administrative rights unless noted.

ACCESS SERVER LS	Allows logical server connection using the SERVER logical server context.
ALTER ANY INDEX	Allows a user to alter and comment on indexes and text indexes on tables and views owned by any user.
ALTER ANY MATERIALIZED VIEW	Allows a user to alter and comment on materialized views owned by any user.
ALTER ANY OBJECT	Allows a user to alter and comment on the following types of objects owned by any user: Data types Events Functions Indexes Materialized views Messages Procedures Sequence generators Spatial reference systems Spatial units of measure Statistics Tables Text configuration objects Text indexes Triggers Views
ALTER ANY PROCEDURE	Allows a user to alter and comment on procedures and functions owned by any user.
ALTER ANY SEQUENCE	Allows a user to alter sequence generators owned by any user.
ALTER ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.
ALTER ANY TABLE	Allows a user to: Alter and comment on tables (including proxy tables) owned by any user. Truncate tables, table partitions, or views owned by any user Comment on tables (including proxy tables) and columns in tables owned by any user.
ALTER ANY TRIGGER	Allows a user to: Alter triggers on tables and views. Issue comments on tables (also requires the ALTER object-level privilege on the table).
ALTER ANY VIEW	Allows a user to alter and comment on views owned by any user.
ALTER DATABASE	Allows a user to: Upgrade a database. Perform cost model calibration. Load database statistics. Alter transaction logs (also requires the SERVER OPERATOR system privilege). Change ownership of the database (also requires the MANAGE ANY MIRROR SERVER system privilege).
ALTER DATATYPE	Allows a user to alter data types.
BACKUP DATABASE	Allows a user to back up a database.
CHECKPOINT	Allows a user to force the database server to execute a checkpoint.
COMMENT ANY OBJECT	Allows a user to comment on any type of object owned by any user that can be created using the CREATE ANY OBJECT system privilege.
CREATE ANY INDEX	Allows a user to create and comment on indexes and text indexes on tables and views owned by any user.
CREATE ANY MATERIALIZED VIEW	Allows a user to create and comment on materialized views owned by any user.
CREATE ANY MUTEX SEMAPHORE	Allows a user to create a mutex or semaphore owned by any user.
CREATE ANY OBJECT	Allows a user to create and comment on the following types of objects owned by any user: Data types Events Functions Indexes Materialized views Messages Procedures Sequence generators Spatial reference systems Spatial units of measure Statistics Tables Text configuration objects Text indexes Triggers Views
CREATE ANY PROCEDURE	Allows a user to create and comment on procedures and functions owned by any user.
CREATE ANY SEQUENCE	Allows a user to create sequence generators, regardless of owner.

CREATE ANY TABLE	Allows a user to: Create and comment on tables (including proxy tables) owned by any user. Comment on columns in tables (including proxy tables) owned by any user.
CREATE ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.
CREATE ANY TRIGGER	Allows a user to create and comment (also requires the ALTER object level privilege on the table) on tables and views.
CREATE ANY VIEW	Allows a user to create and comment on views owned by any user.
CREATE DATABASE VARIABLE	Allows a user to create, select from, update, and drop self-owned database-scope variables.
CREATE DATATYPE	Allows a user to create data types.
CREATE MATERIALIZED VIEW	Allows a user to create and comment on self-owned materialized views.
CREATE MESSAGE	Allows a user to create messages.
CREATE PROCEDURE	Allows a user to create and comment on self-owned procedures and functions. create a self-owned stored procedure or function.
CREATE PROXY TABLE	Allows a user to create self-owned proxy tables.
CREATE TABLE	Allows a user to: Create self-owned tables. Comment on self-owned columns and tables.
CREATE TEXT CONFIGURATION	Allows a user to create and comment on self-owned text configuration objects.
CREATE VIEW	Allows a user to create and comment on self-owned views. Required to create self-owned views.
DEBUG ANY PROCEDURE	Allows a user to debug any database object.
DELETE ANY TABLE	Allows a user to delete rows in tables and views owned by any user.
DROP ANY INDEX	Allows a user to drop indexes and text indexes on tables and views owned by any user.
DROP ANY MATERIALIZED VIEW	Allows a user to drop materialized views owned by any user.
DROP ANY MUTEX SEMAPHORE	Allows a user to drop a mutex or semaphore owned by any user.
DROP ANY OBJECT	Allows a user to drop the following types of objects owned by any user: Data types Events Functions Indexes Materialized views Messages Procedures Sequence generators Spatial reference systems Spatial units of measure Statistics Tables Text configuration objects Text indexes Triggers Views
DROP ANY PROCEDURE	Allows a user to drop procedures and functions owned by any user.
DROP ANY SEQUENCE	Allows a user to drop sequence generators owned by any user.
DROP ANY TABLE	Allows a user to drop tables (including proxy tables) owned by any user.
DROP ANY TEXT CONFIGURATION	Allows a user to drop text configuration objects owned by any user.
DROP ANY VIEW	Allows a user to drop views owned by any user.
DROP DATATYPE	Allows a user to drop data types.
DROP MESSAGE	Allows a user to drop messages.
EXECUTE ANY PROCEDURE	Allows a user to execute procedures and functions owned by any user.
INSERT ANY TABLE	Allows a user to insert rows into tables and views owned by any user.

LOAD ANY TABLE	Allows a user to load data into tables owned by any user.
MANAGE ANY DATABASE VARIABLE	Allows a user to create and drop database-scope variables owned by self or by PUBLIC.
MANAGE ANY DBSPACE	Allows a user to: Create, alter, drop, and comment on dbspaces. Grant and revoke CREATE object-level privileges on dbspaces. Move data to any dbspace. Issue a read-only selective restore statement on any dbspace. Run the database delete file function.
MANAGE ANY EVENT	Allows a user to create, alter, drop, trigger, and comment on events.
MANAGE ANY MIRROR SERVER	Allows a user to: Create, alter, drop, and comment on mirror servers. Change mirror server parameters. Set mirror server options. Change ownership of a database.
MANAGE ANY SPATIAL OBJECT	Allows a user to create, alter, drop, and comment on spatial reference systems and spatial unit of measures.
MANAGE ANY STATISTICS	Allows a user to create, alter, drop, and update database statistics for any table.
MANAGE ANY WEB SERVICE	Allows a user to create, alter, drop, and comment on web services.
MANAGE LISTENERS	Allows a user to start and stop network listeners.
MANAGE MULTIPLEX	Allows users to: Create, alter, drop, or comment on logical servers and logical server policies. Assign a dbspace to logical servers. Release a populated dbspace from the exclusive use of a logical server. Manages failover configurations, and perform a manual failover.
MANAGE PROFILING	Allows a user to manage database server tracing. The DIAGNOSTICS system role is also required to fully utilize diagnostics functionality for user information.
MANAGE ANY PROPERTY HISTORY	Allows a user to turn on and configure the tracking of database server property values.
MONITOR	Allows a user to monitor a database, including accessing privileged statistics, connected users, and locks.
READ CLIENT FILE	Allows a user to read files on the client computer.
READ FILE	Allows a user to read files on the database server computer.
REORGANIZE ANY OBJECT	Allows a user to reorganize tables and materialized views owned by any user.
SELECT ANY TABLE	Allows a user to query tables and views owned by any user.
SELECT PUBLIC DATABASE VARIABLE	Allows a user to select the value of a database-scope variable owned by PUBLIC.
SERVER OPERATOR	Allows a user to: Create, drop, change ownership of a database, and restore the catalog (only). Create, alter, and drop a server. Manage a server cache. Start and stop database or database engine. Encrypt databases. Change a database transaction log.
SET ANY PUBLIC OPTION	Allows a user to set PUBLIC database options that do not require the SET ANY SECURITY OPTION or the SET ANY SYSTEM OPTION system privileges.
SET ANY SYSTEM OPTION	Allows a user to set PUBLIC system database options.
SET ANY USER DEFINED OPTION	Allows a user to set user-defined database options.
SET USER (granted with no administrative rights)	Allows a user to temporarily assume the roles and privileges of another user.

TRUNCATE ANY TABLE	Allows a user to truncate data for tables and materialized views owned by any user.
UPDATE ANY MUTEX SEMAPHORE	Allows a user to update a mutex or semaphore owned by any user.
UPDATE ANY TABLE	Allows a user to update rows in tables and views owned by any user.
UPDATE PUBLIC DATABASE VARIABLE	Allows a user to update database-scope variables owned by PUBLIC.
UPGRADE ROLE	Allows a user to be a default administrator of any system privilege that is introduced when upgrading an SAP IQ database from version 16.0. By default, the UPGRADE ROLE system privilege is granted to the SYS_AUTH_SA_ROLE role, if it exists.
USE ANY SEQUENCE	Allows a user to use sequence generators owned by any user.
VALIDATE ANY OBJECT	Allows a user to validate tables, materialized views, indexes, and text indexes owned by any user.
WRITE CLIENT FILE	Allows a user to write files to the client computer.
WRITE FILE	Allows a user to write files on the database server computer.

SYS_AUTH_SSO_ROLE role is granted the following privileges:

All grants are with administrative rights unless noted.

ACCESS USER PASSWORD	Allows a user to access views that contain password hashes, and perform operations that involve accessing passwords, such as unloading, extracting, or comparing database
ALTER ANY OBJECT OWNER	Allows a user to alter the owner of any type of table object. This privilege does not allow changing of the owner of other objects, such as procedures, materialized views, and so on.
CHANGE PASSWORD	Allows a user to manage user passwords for any user. This system privilege can apply to all users, or it can be limited to a set of specified users, or users who are granted one or more specified roles. This system privilege is not required to change a user's own password.
DROP CONNECTION	Allows a user to drop any connections to the database.
MANAGE ANY LDAP SERVER	Allows a user to create, alter, drop, validate, and comment on LDAP servers.
MANAGE ANY LOGIN POLICY	Allows a user to create, alter, drop, and comment on login policies.
MANAGE ANY OBJECT PRIVILEGE	Allows a user to: Grant and revoke SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES, LOAD, and TRUNCATE object-level privileges on tables owned by any user. Grant and revoke SELECT, INSERT, DELETE, and UPDATE object-level privileges on views owned by any user. Grant and revoke EXECUTE privileges on procedures and functions owned by any user. Grant and revoke USAGE object-level privileges on sequence generators owned by any user. Grant and revoke CREATE object-level privileges on dbspaces.
MANAGE ANY USER	Allows a user to: Create, alter, drop, and comment on database users (including assigning an initial password). Force a password change on next login for any user. Assign and reset the login policy for any user. Create, drop, and comment on integrated logins and Kerberos logins. Create and drop external logins.

MANAGE AUDITING	Allows a user to run the sa_audit_string stored procedure.
MANAGE CCERTIFICATES	Allows a user to create, alter, drop, and comment on certificates.
MANAGE ROLES	Allows a user to create new roles and act as a global administrator for new and existing roles. By default, MANAGE ROLES is granted administrative rights on each newly created role. A user requires administrative rights on the role to delete it. Administration of a role can also be granted directly to users either during or after the creation of the role. When granted directly to a user, the user does not require the MANAGE ROLES system privilege to administer the role. If no role administrator is specified during role creation, the MANAGE ROLES system privilege is automatically granted to the role with the ADMIN ONLY OPTION clause, allowing the global role administrator to administer the role. If at least one role administrator is specified during creation, the MANAGE ROLES system privilege is not granted to the role, and global role administrators cannot manage the role.
OFLINE RESET PASSWORD (granted with administrative only rights)	Allows the user to reset another user's password when using the database server -orp option. The user to perform the reset should not be the DBA user.
SET ANY SECURITY OPTION	Allows a user to set any PUBLIC security database options.
SET USER (granted with administrative only rights)	Allows a user to temporarily assume the roles and privileges of another user. This system privilege can apply to all users, or can be limited to a set of specified users, or users who are granted one or more specified roles.

3.1.1.2 Changing the DBA User Password

Change this password to prevent unauthorized access to your database.

Prerequisites

- Requires the CHANGE PASSWORD system privilege.

Context

DBA is a highly privileged user and you should have its passwords regularly. By default, the DBA password never expires.

The password must confirm to the restrictions outlined in Resetting the DBA Password (Command Line) [page 19]

A user granted the `CHANGE PASSWORD` system privilege can change the DBA user's password provided the DBA user is not subject to the dual control password login policy. When the DBA user has the `CHANGE_PASSWORD_DUAL_CONTROL` option enabled in their login policy, then a password change requires two distinct users granted the `CHANGE PASSWORD` system privilege to complete the change. The dual control method is the preferred approach for recovering lost DBA passwords. Neither method requires stopping the database.

A user granted the `CHANGE PASSWORD` privilege can change the DBA user's password unless the DBA user is not subject to the dual control password login policy. If the DBA user has the CHANGE_PASSWORD_DUAL_CONTROL option enabled in their login policy, then a password change requires two different users granted the CHANGE PASSWORD privilege to complete the change. This is the preferred method for handling lost passwords. It does not require that the database be stopped.

If the DBA password is lost, take the server offline and follow the steps in Resetting the DBA Password (Command Line) [page 19] .

Procedure

To change the DBA password, execute:

```
ALTER USER DBA IDENTIFIED BY <password>;
```

Related Information

ALTER USER Statement

3.1.1.3 Resetting the DBA Password (Command Line)

Reset the DBA user's password if the password is lost.

Prerequisites

- You know the user ID and password of a database user that has been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.
- You have access to the database server.
- You have read/write access to the database file.
- The database is stopped.

Context

The offline password reset task can be used to reset any user's password including those users subject to the dual control password login policy.

Procedure

1. Start the database server with the -orp option and include a new password for the DBA user with the lost password. The specified user must have been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.

   A message appears indicating whether the password reset was successful and then the database server shuts down.
2. Restart the database.

Example

In this example, user `user1`, who has the SYS_OFFLINE_RESET_PASSWORD_ROLE system role granted, resets the `DBA` user's password:

```
iqsrv16 -orp "UID=DBA;NEWPWD=newpassword12345;AUTHUID=user1;AUTHPWD=pwd123456"
mydb.db
```

In this section:

3.1.1.3.1 -orp Database Server Option

Resets the password of the specified user.

Syntax

```
iqsrv16 -orp { UserID | UID }=<user-ID>;{ NewPassword | NEWPWD }=<new-password>;
{ AuthUserID | AUTHUID }=<auth-user-ID>;{ AuthPassword | AUTHPWD }=<auth-
password>" <database-file>
```

On UNIX and Linux, quotation marks are required for the -orp value when semicolons are used.

Note

SAP recommends you use the `start_iq` utility for server startup operations. However, the `-orp` database server option is an `iqsrv16` command line argument, not a `start_iq` argument. Attempting to use `-orp` in a `start_iq` command will result in a syntax error.

Applies to

All operating systems.

Permissions

SYS_OFFLINE_RESET_PASSWORD_ROLE system role

SERVER OPERATOR system privilege

Allowed Values

user-ID

The user whose password is being reset.

new-password

The new password for `<user-ID>`.

auth-user-ID

The user who has been granted the SYS_OFFLINE_RESET_PASSWORD_ROLE system role by the role administrator.

auth-password

The password for `<auth-user-ID>`.

database-file

The database that contains the user whose password you are resetting.

Remarks

You must specify one — and only one — database file with -orp.

The `<user-ID>` user (normally the DBA) whose password you are resetting must have a password. Password reset is not allowed for users without a valid password.

The new password must have a length of at least six characters.

You cannot specify any other database server options with the -orp option, with the exception of -o, -ep, and -ek.

You must shut down the database server before restarting it with the -orp option. When you specify -orp, the database server starts and attempts to reset the password of the specified user. The database server shuts down immediately after the password reset is attempted. After the shutdown, a message appears indicating that the password reset was successful or showing the reason that the reset failed.

> **Note**
>
> The database user that starts the database sever with the -orp option must have the SYS_OFFLINE_RESET_PASSWORD_ROLE system role.

Example

The password for the user `DBA` is lost. The user reset_user has the SYS_OFFLINE_RESET_PASSWORD_ROLE system role and resets the `DBA` user's password to be **`newpassword`**:

```
iqsrv16 -orp "UID=DBA;NEWPWD=newpassword;AUTHUID=reset_user;AUTHPWD=sql456"
mydb.db
```

Related Information

Granting the SYS_OFFLINE_RESET_PASSWORD_ROLE System Role [page 102]
Resetting the DBA Password (Command Line) [page 19]

3.1.1.4 Create a DBA Equivalent User

Create a user with equivalent privileges to the DBA user.

Prerequisites

- A designated user to act as an equivalent user exists. See Creating a New User [page 24].

Context

As a highly privileged user, DBA shouldn't be used on a daily basis to administer the database. For security purposes, SAP recommends that you create a replacement administrative user with privileges equivalent to DBA and then lock the DBA user.

Granting a user the `SYS_DL_CUSTOMER_ADMIN_ROLE` makes the user equivalent in privilege to DBA. Include the `WITH ADMIN` option to allow the user to grant the role to other users if needed.

Granting a user the SYS_AUTH_DBA_ROLE role makes the user equivalent in privilege to DBA. Include the `WITH ADMIN OPTION` option to allow the user to grant the role to other users if needed.

Procedure

1. Log in as the DBA user.

2. Grant the designated user the SYS_AUTH_DBA_ROLE role with administrative privilege.

```
GRANT ROLE  SYS_AUTH_DBA_ROLE TO <user_name> WITH ADMIN OPTION;
```

3. Verify that you can login as the new equivalent user.

Related Information

GRANT ROLE Statement

3.1.1.5 Locking the DBA User

Locking the DBA user enhances database security by preventing daily administrative use of the DBA account

Prerequisites

- A user granted equivalent privileges to DBA exists. See Create a DBA Equivalent User [page 22].
- A login policy with the LOCK option set to ON exists. See Login Policy Options [page 137].

.

Context

Because DBA is the well-known default administrative user, it is a prime target for brute-force and credential-stuffing attacks. To improve security and accountability, SAP recommends that you:

1. Create a new administrative user and grant it all the rights and privileges of DBA (see Create a DBA Equivalent User [page 22]).
2. Lock the DBA user once the equivalent user is in place (see Locking the DBA User [page 23]).

> ⚠ Caution
>
> Verify that you can log in as the equivalent user and it is granted the SYS_AUTH_DBA_ROLE role with administrative privileges **before** locking the DBA user.

Procedure

1. Connected as the equivalent user, assign the DBA user a login policy with the `LOCK` option set to `ON`.

```
ALTER USER DBA LOGIN POLICY <policy_name>
```

2. Verify the you can no longer log in as DBA.

Related Information

ALTER USER Statement

3.1.2 Creating a New User

Create a new user ID.

Prerequisites

- Requires the MANAGE ANY USER system privilege.

Context

When creating a user, follow best practices to ensure optimal security. See User ID and Password Restrictions and Considerations [page 145]

If not specified, the login policy ROOT (default) is assigned. See Login Policy Options [page 137].

If you are implementing the Dual Control Password Management Option [page 151], ensure the user is assigned a login policy with the following ;login policy options defined:

- CHANGE_PASSWORD_DUAL_CONTROL option is enabled.
- MAX_FAILED_LOGIN_ATTEMPTS
- MAX_DAYS_SINCE_LOGIN

Procedure

To create a new user, execute:

```
CREATE USER <user-name> IDENTIFIED BY <password>
[ LOGIN POLICY <policy-name> ]
```

To change the assigned login policy afterwards, see Assigning a Login Policy to an Existing User [page 143].

Example

This statement creates user ID user1 with password wElc0me33:

```
CREATE USER Joe IDENTIFIED BY wElc0me33
```

Related Information

CREATE USER Statement

3.1.3 Changing a User's Password

Change the password of another user.

Prerequisites

- No additional privilege is required to change your own password.
- CHANGE PASSWORD system privilege is required to change another user's password.

Context

If changing another user's password, then include the `FORCE PASSWORD CHANGE ON` clause in the `ALTER USER` statement. The next time the user logs in, they are prompted to change their password. This ensures that no one but the user knows their password.

See User ID and Password Restrictions and Considerations [page 145] for guidance on defining new passwords.

Procedure

To change a user password, execute:

```
ALTER USER <user_ID> IDENTIFIED BY <password> FORCE PASSWORD CHANGE ON
```

Example

This statement assigns the new password P&ssW0rd to user1:

```
ALTER USER user1 IDENTIFIED BY P&ssW0rd
```

Related Information

ALTER USER Statement

3.1.4 Locking a User Account

To permanently lock a user account, you must assign a login policy with the locked option set to ON to the account. Once disabled, a user cannot connect to the SAP IQ server.

Prerequisites

- A login policy with the `LOCKED` option set to `ON` exists. See Login Policies [page 136]
- Requires the `MANAGE ANY USER` system privilege.

Context

You cannot specify multiple user names in a single `ALTER USER` statement when assigning a login policy to users. A user can only be assigned one login policy.

Procedure

Execute:

```
ALTER USER <userID> LOGIN POLICY <policy_name>
```

Example

This statement assigns the `locked_users` login policy to user1. User1 can no longer log in.

```
ALTER USER user1 LOGIN POLICY locked_users;
```

Related Information

ALTER USER Statement

3.1.5 Unlocking a User Account

Unlock a permanently or temporarily locked user account.

Prerequisites

- Requires the MANAGE ANY USER system privilege.

Context

A user account can be temporarily or permanently. A temporary lock occurs when the user exceeded the failed number of logins allowed or exceeded the maximum number of days since the last login. A permanent lock occurs when the user is assigned a login policy with the `LOCKED` option set to `ON`. The unlock process depends on the type of lock.

Forcing the reset of the login policy reverts the settings of the user's login to the original values in the login policy. This clears all locks implicitly set due to the user exceeding the failed number of logins or exceeding the maximum number of days since the last login. Resetting the values in a user's login policy does not reset the values for all users assigned the same login policy.

For options to automatically unlock temporarily locked user accounts, see Automatic Unlocking of User Accounts [page 143].

Procedure

1. To unlock a temporarily locked account, execute:

   ```
   ALTER USER <user-name> RESET LOGIN POLICY;
   ```

2. To unlock a permanently locked account, reassign the user to a login policy with the `LOCKED` option set to `OFF`.

```
ALTER USER <user-name> LOGIN POLICY <login-policy-name>;
```

Example

Assuming that the `LOCKED` option in login policy `allow_login`is set to `OFF`, this example replaces the login policy currently assigned to user1 with login policy `allow_login`:

```
ALTER USER user1 LOGIN POLICY allow_login;
```

Assuming user1 either exceeded the `MAX_FAILED_LOGIN_ATTEMPTS` or `MAX_DAYS_SINCE_LOGIN` values, which locked the account, this example forces the reset of the values for user1's account only:

```
ALTER USER user RESET LOGIN POLICY;
```

Related Information

ALTER USER Statement

3.1.6 Listing Roles and System Privileges Granted to a User

Display the roles and system privileges granted directly or inherited by a user.

Prerequisites

Requires one of the following:

- You are listing system privileges and roles granted to self.
- `MANAGE ROLES` system privilege

Context

The `sp_displayroles` stored procedure returns the system privileges and roles granted to a specified user. You can show only direct grants (the default) or use the `EXPAND_DOWN` parameter to show both direct and inherited grants. The output also indicates whether each role or system privilege was granted with administrative rights. If no user name is specified, the procedure returns only the direct grants of the connected user.

Procedure

To display both direct and inherited grants for a user, execute:

```
CALL sp_displayroles('<user-nname>','EXPAND_DOWN');
```

Sample Output:

```
CALL sp_displayroles('<user1>','EXPAND_DOWN');
```

	role_name	parent_role_name	grant_type	role_level
1	ROLE_2	NULL	ADMIN	1
2	PUBLIC	NULL	NO ADMIN	1
3	ALTER ANY MATERIALIZED VIEW	NULL	ADMIN	1
4	ROLE_3	ROLE_2	NO ADMIN	2
5	ALTER ANY VIEW	ROLE_2	NO ADMIN	2
6	ALTER ANY SEQUENCE	ROLE_3	ADMIN	3

Where:

- A `parent_role_name` value of `NULL` with a `<role_level>` value of 1 means that a direct grant.
- A `parent_role_name` value of a role name with a `role_level` value greater than 1 means an inheritance from the parent role rather than a direct grant.
- The `grant_type` value indicates whether the grant or inheritance was with or without administrative privilege.

Summary of output:

Row 1

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that `user1` is directly granted membership in the `ROLE_2` role. The `<grant_type>` of `ADMIN` means that `user1` has administrative rights on the `ROLE_2` role.

Row 2

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that `user1` is directly grant membership in the `PUBLIC` role . The `<grant_type>` value of `NO ADMIN` means that `user1` doesn't have administrative rights on the `PUBLIC`role.

Row 3

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that `user1` is directly grant the `ALTER ANY MATERIALIZED VIEW` system privilege. The `<grant_type>` value of `ADMIN` means that `user1` has administrative rights on the system privilege.

Row 4

A `<parent_role_name>` value of `ROLE_2` and a `<role_level>` value of 2 means that `user1` inherits membership in `ROLE_3` through its membership in `ROLE_2`. The `<grant_type>` of `NO ADMIN` means that `user1` doesn't have administrative rights on the `ROLE_3` role.

Row 5

A <parent_role_name> value of ROLE_2 and a <role_level> value of 2 means that user1 inherits the grant of the ALTER ANY VIEW system privilege through its membership in ROLE_2. The <grant_type> of NO ADMIN means that user1 doesn't have administrative rights on the ALTER ANY VIEW system privilege.

Row 6

A <parent_role_name> value of ROLE_3 and a row_level value of 3 means that user1 inherits the grant of the ALTER ANY SEQUENCE system privilege in ROLE_3 through membership in ROLE_2. The <grant_type> of ADMIN means that user1 has administrative privilege on the ALTER ANY SEQUENCE system privilege, even if user1 doesn't have administrative rights on the ROLE_3 role itself.

Related Information

sp_displayroles System Procedure

3.1.7 Listing Object Privileges Granted to a User

List the object privileges granted directly or indirectly by a user.

Prerequisites

- You are listing object privileges granted to self.
- MANAGE ANY OBJECT PRIVILEGE system privilege

Context

The results for sp_objectpermission include privileges explicitly granted to the user and inherited object privileges. The grantee column indicates the recipient of the object privilege or where the object privilege was inherited from. The grantor column indicates who performed the actual granting. The grantable column tells you whether the object privilege was granted with administrative rights.

> → Tip
>
> By default, all users are granted membership in the PUBLIC role and granted EXECUTE object privilege on the role. This results in a long list of granted object privileges, making it difficult to find object privileges granted outside the scope of the PUBLIC role. To display a more manageable list, filter the list by excluding grants by grantor SYS.

Procedure

To display the object level privileges granted to a user, execute:

```
SELECT * FROM sp_objectpermission( '<user-name>' ) WHERE GRANTOR <> 'SYS';
```

Sample Output:

```
SELECT * FROM sp_objectpermission( 'user2' ) WHERE GRANTOR <> 'SYS';
```

	grantor	grantee	object_name	owner	object_type	column_name	permission	grantable
1	USER1	user2	table1	user1	TABLE	NULL	SELECT	N
2	USER1	user2	pse1	user1	PSE	NULL	ALTER	Y
3	USER1	user2	my_schema1	user1	SCHEMA	NULL	SELECT	N
4	USER1	role1	table1	user1	COLUMN	col1	DELETE	N
5	USER1	role1	proc1	user1	PROCEDURE	NULL	EXECUTE	N
6	USER1	role2	table1	user1	TABLE	NULL	UPDATE	Y

Summary of Output:

Row 1 The `SELECT` privilege on table `table1` has been granted to `user2` with no administrative rights.

Row 2 The `ALTER` privilege on PSE `pse1` has been granted to `user2` with administrative rights.

Row 3 The `SELECT` privilege on schema `my_schema1` has been granted to `user2` with no administrative rights.

As a member of `role1`, `user2` inherits the following object privileges from `role1`:

Row 4 The `DELETE` privilege on column `col1` of table `table1` has been inherited by `user2` with no administrative rights on the column.

Row 5 The `EXECUTE` privilege on procedure `proc1` has been inherited by `user2`with no administrative rights.

`role2` has been granted to `role1`. Since `user2` is a member of `role1`, `user2` inherits the following object privileges from `role2`:

Row 5 The `UPDATE` privilege on table `table1` has been inherited by `user2` with administrative rights on the `UPDATE` privilege.

Related Information

sp_objectpermission System Procedure

3.1.8 Deleting a User

Remove a user ID from the database.

Prerequisites

- Requires the MANAGE ANY USER system privilege.
- The user being deleted does not own any database objects and is not currently connected to the database.

Context

If the user being deleted has any external logins defined, the external logins are deleted as part of the process. However, any related objects on remote servers, such as proxy users or remote login mappings, are not removed. When dropping a user, any privileges and roles granted by this user are also removed. If the user being deleted owns any objects in the database, the following error message appears, and the command fails:

```
Cannot drop a user that owns tables in runtime system SQLCODE=-128, ODBC 3
State="42000"
    Line 1, column 1
```

Procedure

Execute:

```
DROP USER <userID>
```

Example

This statement drops user ID `user1` from the database:

```
DROP USER user1;
```

Related Information

DROP USER Statement

3.2 Managing User Connections

There are several ways to manage user connections.

You can:

- Limit the number of active logins for a single user – assign user to a login policy in which the `MAX_CONNECTIONS` login policy option is set.
- Lock a user account:
  - Explicitly – assign user to a login policy in which the `LOCKED` option is set to ON.
  - Implicitly – assign user to a login policy in which the `MAX_FAILED_LOGIN_ATTEMPTS` option is set. If the user exceeds the value when attempting to log in, his or her user account is locked.
- Set a password expiry condition – assign user to a login policy in which the `PASSWORD_EXPIRY_ON_NEXT_LOGIN` login policy option is set. You can also execute the `CREATE USER` or `ALTER USER` statements, including the `FORCE PASSWORD CHANGE` clause.

Assigning a login policy to a user, or forcing a password change requires the `MANAGE ANY USER` system privilege. Creating or altering a login policy requires the `MANAGE ANY LOGIN POLICY` system privilege.

Related Information

Users [page 11]
Login Policies [page 136]

3.3 Preventing Connection After Failed Login Attempts

Prevent a user from connecting after exceeding the maximum failed login attempts.

Prerequisites

- The MANAGE ANY LOGIN POLICY system privilege to create or alter the login policy.
- The MANAGE ANY USER system privilege to assign the login policy to users.

Context

You can set the system can be set to automatically lock an account if a user fails to enter valid login credentials after a specified number of attempts. Once locked, the user cannot connect, even if valid credentials are subsequently entered; the account remains locked until it is manually unlocked. The

MAX_FAILED_LOGIN_ATTEMPTS login policy option controls the number of sequential failed attempts before the user account is locked. You can set this value in a new or existing login policy, including the root login policy, and it then applies to all users who are assigned the login policy.

Procedure

1. In a new or existing login policy execute:

   ```
   CREATE LOGIN POLICY lp MAX_FAILED_LOGIN_ATTEMPTS=<numer>
   ```

   ```
   ALTER LOGIN POLICY lp MAX_FAILED_LOGIN_ATTEMPTS=MAX_FAILED_LOGIN_ATTEMPTS
   ```

2. Assign the login policy to applicable users, as needed.

Example

This example creates a new login policy named `lp`, which automatically locks a user account after 5 failed attempts:

```
CREATE LOGIN POLICY lp max_failed_login_attempts=5
```

This example modifies an existing login policy named `exist_lp`, which automatically locks a user account after 5 failed attempts:

```
ALTER LOGIN POLICY lp max_failed_login_attempts=5
```

This example assigns the login policy `lp` to user `John`. Once `John` is assigned the `lp` login policy, he cannot log in if he enters invalid credentials five times in sequence.

```
ALTER USER John LOGIN POLICY lp
```

Related Information

Login Policies [page 136]
CREATE LOGIN POLICY Statement
ALTER LOGIN POLICY Statement
Multiplex Login Policy Configuration

3.4 Manage Connections Using Stored Procedures

There are several stored procedures for managing user connections.

This table lists the procedure available to perform each SAP IQ login management function.

Stored Procedure	Purpose	System Privilege Required
sa_get_user_status System Procedure	Retrieve the current status of all existing users	MANAGE ANY USER system privilege to retrieve the current status of all existing users. Users without the MANAGE ANY USER system privilege can retrieve only their current status.
sp_expireallpasswords System Procedure	Immediately expire all user passwords	MANAGE ANY USER system privilege
(Deprecated) sp_iqaddlogin Procedure	Add users, define their passwords, specify login policy, and password expiry on next login	MANAGE ANY USER system privilege
sp_iqcopyloginpolicy Procedure	Create a new login policy by copying an existing one	MANAGE ANY LOGIN POLICY system privilege
(Deprecated) sp_iqdroplogin Procedure	Drop the specified user	MANAGE ANY USER system privilege
(Deprecated) sp_iqmodifylogin Procedure	Assign a given user to a login policy	MANAGE ANY USER system privilege
sp_iqmodifyadmin Procedure	Set an option on a named login policy to a certain value	MANAGE ANY LOGIN POLICY system privilege
(Deprecated) sp_iqpassword Procedure	Change your own or another user's password	All users can run `sp_iqpassword` to change their own passwords. CHANGE PASSWORD system privilege is required to change the password of another user.

3.5 Manage Resources Used by Connections

Building a set of users and roles allows you to manage permissions on a database. Another aspect of database security and management is to limit the resources an individual user can use.

For example, you may want to prevent a single connection from taking too much available memory or CPU resources, and slowing down other database users.

Database Options That Govern User Resources

Database options that control resources are called resource governors. These options are set using the `SET OPTION` statement.

Option	Description
CURSOR_WINDOW_ROWS	Defines the number of cursor rows to buffer.
MAX_CARTESIAN_RESULT	Limits the number of result rows from a query containing a Cartesian join.
MAX_IQ_THREADS_PER_CONNEC-TION	Sets the number of processing threads available to a connection for use in IQ operations.
QUERY_TEMP_SPACE_LIMIT	Limits the amount of temporary space available to any one query before it is rejected.
QUERY_ROWS_RETURNED_LIMIT	Tells the query optimizer to reject queries that consume too many resources. If the optimizer estimates that the result set from the query exceeds the value of this option, then the optimizer rejects the query and returns an error message.

The following database options affect the SAP IQ, but have limited impact on the instance:

Option	Description
MAX_CURSOR_COUNT	Limits the number of cursors for a connection.
MAX_STATEMENT_COUNT	Limits the number of prepared statements for a connection.

3.6 Impersonation

A user may temporarily impersonate another user's roles and system privileges to execute operations, as long as the impersonating user already has the minimum required privileges (at-least criteria) for the task.

Example Scenario

`User1` normally performs a critical task but is unavailable. `User2` can perform the task, but also has additional privileges that may change the task outcome. To ensure consistent behavior, `User2` impersonates User1 and executes the task under `User1's` privilege set.

How Impersonation Works

1. Grant the SET USER system privilege to the impersonating user.
2. Execute the SETUSER statement to start impersonation.

> **Note**
>
> SET USER (two words) is the system privilege
>
> SETUSER (one word) is the SQL statement.

During impersonation, audit logs record the impersonated user ID for actions performed. The SETUSER action itself is also audited, so you can determine that impersonation occurred and identify who initiated it.

In a multiplex configuration, if an active impersonation exists on a coordinator connection and a grant or revoke would violate the at-least criteria, that connection is terminated. Because terminating the connection ends impersonation, the violation condition is removed, and the `GRANT or REVOKE` statement then executes successfully.

Scope Options When Granting SET USER

- Any user in the database.
- Any user within a specified list of users (`<target_users_list>`).
- Any user who is a member of one or more of the specified roles (`<target_roles_list>`).

At-Least Criteria

To impersonate a target user, the impersonating user must have, at minimum:

- All roles granted to the target user
- All system privileges granted to the target user
- The same or higher administrative options

This requirement is called the **at-least criteria**.

The impersonating user may have additional roles, privileges, or higher administrative levels, but not fewer.

Behavior During Active Impersonation

- Grants or revokes are allowed only if the at-least criteria remains satisfied.
- If a grant or revoke causes a violation, the statement fails with an error.

Example

- USER1 is impersonating USER2.
- If a new role is granted to USER1 only, the grant succeeds because User1 still has at least what USER2 has.
- If a new role is granted to USER2 but not USER1, the statement fails because USER2 would then have more than User1.

In a multiplex configuration, if an impersonation is active in a connection that is present in the coordinator, and an attempt is made to grant or revoke roles and privileges that violates the at-least criteria, the connection containing the active impersonation terminates. Since terminating the connection also terminates the impersonation, violation of at-least criteria is no longer an issue, and the `GRANT` or `REVOKE` statement executes successfully.

In this section:

Remove the ability of a user to impersonate other users, and to administer the SET USER system privilege.

3.6.1 Requirements for Impersonation

A user can successfully impersonate another user only if a specific set of criteria is met, also called the at-least requirements.

There are four criteria to successful impersonation:

1. The impersonator has been granted the right to impersonate the target user.
2. The impersonator has, at minimum, all the roles and system privileges granted to the target user.
3. The impersonator has been granted the said roles and system privileges with similar or higher administrative rights.

> **Note**
>
> For the purposes of meeting at-least administrative rights criteria, the WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are considered to grant similar administrative rights. They are also considered to grant higher administrative rights than the WITH NO ADMIN OPTION clause.

4. If the target user has been granted a system privilege that supports extensions, the clauses used to grant the system privilege to the impersonator are a super-set of those used for the target user. Only the SET USER and CHANGE PASSWORD system privileges support extensions.
   - The ANY clause is considered a super-set of the `<target_roles_list>` and `<target_users_list>` clauses. If the target user has been granted the SET USER system privilege with an ANY grant, the impersonator must also have the ANY grant.
   - If the target user has been granted the SET USER system privilege with both the `<target_roles_list>` and `<target_users_list>` clauses, the impersonator must also have been granted the system privilege with the two clauses, and the target list of each clause must be equal to, or a super set of, the corresponding clause grant of the target user. For example, if the target lists of both the impersonator and target user contain `User1`, `User2` and `Role1`, `Role2`, respectively, the target list grants for each clause are said to be equal. Alternately, if the target list grants of the impersonator contain `User1`, `User2`, and `Role1`, `Role2`, respectively, while the target list grants of the target user contain `User1` and `Role2` only, the target list grants of the impersonator are said to be a super-set of the target user.
   - If the target user has been granted the SET USER system privilege with a single target list clause, the target list of the impersonator must be equal to or a super-set of the list of the target user. For example, the `<target_user_list>` of both the impersonator and the target user contain `User1` and `User2` (equal) or the impersonator list contains `User1`, `User2`, while the target user contains `User2`; `User1`, `User2` (impersonator list) is a super-set of `User2` (target user list).
   - By definition, a user can always impersonate himself or herself. Therefore, if the target user is granted the right to impersonate the impersonator, this does not violate the equal to or a super-set of criteria requirement of the impersonator. For example, `User3` is the impersonator and `User4` is the target user. The `<target_user_list>` for `User3` contains `User4` and `User5`. The `<target_user_list>` for `User4` contains `User3` and `User5`. If you remove the impersonator from the target list, the target list of `User3` meets the criteria requirement.

Scenario 1

Assuming that criteria 2 and 3 are met, consider the following scenario:

- There are five users: <User1>, <User2>, <User3>, <User4>, and <User5>.
- There are two roles: <Role1> and <Role2>.
- <User1> has been granted the SET USER system privilege with the ANY clause.
- <User2> has been granted the SET USER system privilege with the <target_users_list> clause for <User1> and <User4>.
- <User3> has been granted the SET USER system privilege with the <target_users_list> clause for <User1>, <User2>, <User4> and, <User5>, and the ANY WITH ROLES <target_roles_list> clause for <Role1> and <Role2>.
- <User4> has been granted the SET USER system privilege with the ANY clause and the <target_roles_list> clause for <Role1>.
- <User5> has been granted the SET USER system privilege with the <target_users_list> clause for <User4> and the ANY WITH ROLES <target_roles_list> for <Role1>.

<User1> and <User4> can successfully impersonate <User2>, <User3>, and <User5> because each is granted the SET USER system privilege with the ANY clause (criteria 4).

<User1> and <User4> can impersonate each other because they each have the ANY grant (criteria 4).

<User2>, <User3>, and <User5> cannot impersonate <User1> or <User4> because they do not have the ANY grant (criteria 4).

<User2> cannot impersonate <User3> or <User5> because:

- <User2> is not granted the right to impersonate these users (criteria 1).
- The SET USER system privilege is not granted to <User2> with the <target_roles_list> clause (criteria 4).

<User3> can successfully impersonate <User2> because:

- <User3> is granted the right to impersonate <User2> via the <target_users_list> clause (criteria 1).
- The <target_users_list> clause for <User3> is a super-set of <User2> (criteria 4). Though <User3> has a grant with the <target_role_list> clause, it is not required to satisfy the requirements for impersonation of <User2> because the latter does not have the same grant.

<User3> can successfully impersonate <User5> because:

- <User3> is granted the right to impersonate <User5> via the <target_users_list> clause (criteria 1).
- The <target_users_list> clause list for <User3> is a super-set of <User5> (criteria 4).
- The <target_roles_list> clause lists for <User3> and <User5> are equivalent (criteria 4).

<User5> cannot impersonate any other user because:

- <User1> and <User4> have an ANY grant (Criteria 4).
- <User2> and <User3> have a grant with a <target_users_list> clause that is not a sub-set of the grant to <User5> (criteria 4).
- <User3> has a grant with a <target_roles_list> clause that is not a subset (criteria 4).

Scenario 2

Assuming that criteria 1 and 4 are met, consider the following:

- There are two users: <User6> and <User7>.
- There are two roles: <Role4> and <Role5>.
- <User6> has been granted <Role4> with the WITH ADMIN OPTION clause, <Role5> with the WITH ADMIN ONLY OPTION clause, and the MANAGE ANY USER system privilege with the WITH ADMIN OPTION clause.
- <User7> has been granted <Role4> with the WITH ADMIN OPTION clause and <Role5> with the WITH NO ADMIN OPTION clause.

<User6> can successfully impersonate <User7> because:

- Both <User6> and <User7> are granted <Role4> and <Role5>. It does not matter that <User6> is granted additional privileges (MANAGE ANY USER system privilege) (criteria 2).
- <User6> is granted <Role4> with equivalent administrative rights as <User7>. <User6> is granted <Role5> with higher administrative rights than <User7> (criteria 3).

<User7> cannot impersonate <User6> because:

- <User7> is granted <Role4> and <Role5>, but not the MANAGE ANY USER system privilege (criteria 2).
- <User7> is granted <Role5> with lower administrative rights than <User6> (criteria 3).

Scenario 3

Consider the following:

- There are three users: <User8>, <User9> and <User10>.
- There are two roles: <Role5> and <Role6>.
- <User8> has been granted <Role5> with the WITH ADMIN OPTION clause, and the MANAGE ANY USER system privilege with the WITH ADMIN OPTION clause.
- <User9> and <User10> has been granted <Role5> with the WITH NO ADMIN OPTION clause.
- <User8> has been granted the SET USER system privilege to impersonate <User9> and <User10> with the <target_users_list> clause.
- <User9> as been granted the SET USER system privilege to impersonate <User10> with the <target_users_list> clause.

<User8> can successfully impersonate <User9> because:

- <User8> is granted the right to impersonate <User9> via the <target_users_list> clause (criteria 1).
- The <target_users_list> clause list for <User8> is a super-set of <User9> (criteria 4).
- Both <User8> and <User9> are granted <Role5>, with <User8> granted higher administrative rights to the role than <User9> (criteria 2 and 3).

<User8> can successfully impersonate <User10> because:

- <User8> is granted the right to impersonate <User10> (Criteria 1).
- Since <User10> is not granted the SET USER system privilege, requirement 4 is not applicable.

- Both <User8> and <User10> are granted <Role5>, with the same administrative rights to the role (criteria 2 and 3).

<User9> cannot impersonate <User8> because:

- <User9> is not granted the right to impersonate <User8> (Criteria 1.)
- Though both <User8> and <User9> are granted <Role5>, the grant for <User9> is with less administrative rights to the role than for <User8> (criteria 3).

Criterion are validated occurs when the SETUSER statement is executed, not when the SET USER system privilege is granted. If a user fails to meet any of the criteria when the SETUSER statement is issued, a `permission denied` message appears, and the impersonation does not begin.

3.6.2 Granting the SET USER System Privilege to a User

Allow one user to impersonate another user in the database. The system privilege can be granted with or without administrative rights.

Prerequisites

- Requires the SET USER system privilege granted with administrative rights.
- Each target user specified (target_users_list) must be an existing user or user-extended role with a login password.
- Each target role specified (target_roles_list) must be an existing user-extended or user-defined role.

Context

You can grant a user the ability to impersonate any user in the database (ANY), only specific users (<target_users_list>), or members of specific roles (ANY WITH ROLES <target_roles_list>). Administrative rights to the SET USER system privilege can be granted only when using the ANY clause.

If no clause is specified, ANY is the default.

When regranting the SET USER system privilege to a user, the effect of the grant is cumulative.

If no administrative clause is specified when using the ANY clause, WITH NO ADMIN OPTION is the default.

WITH NO ADMIN OPTION is the only valid administrative clause with the <target_users_list> or <target_roles_list> clauses.

Procedure

To grant the SET USER system privilege, execute one of these statements:

Grant Type	Statement
System privilege to impersonate any database user with full administrative rights	`GRANT SET USER (ANY) TO <user_ID [,...]> WITH ADMIN OPTION`
System privilege to impersonate any database user, with administrative rights only	`GRANT SET USER (ANY) TO <user_ID [,...]> WITH ADMIN ONLY OPTION`
System privilege to impersonate any database user, with no administrative rights	`GRANT SET USER (ANY) TO <user_ID [,...]> WITH NO ADMIN OPTION`
System privilege to impersonate specified users	`GRANT SET USER (<target_users_list>) TO <user_ID [,...]>`
System privilege to impersonate any member of specified roles	`GRANT SET USER (ANY WITH ROLES <target_roles_list>) TO <user_ID [,...]>`
System privilege to impersonate specified users and members of specified roles	`GRANT SET USER (<target_users_list>), (ANY WITH ROLES <target_roles_list>) TO <user_ID [,...]>`

Example

Both statements grant <Sam> the ability to impersonate any database user:

```
GRANT SET USER (ANY) TO Sam
or
GRANT SET USER TO Sam
```

This statement grants <Bob> and <Jeff> the ability to impersonate <Mary>, <Joe>, or <Sue> only.

```
GRANT SET USER (Mary, Joe, Sue) TO Bob, Jeff
```

This statement grants <Mary> the ability to impersonate any member of the <Sales1> role:

```
GRANT SET USER (ANY WITH ROLES Sales1) TO Mary
```

This statement grants <Sarah> the ability to impersonate <Joe> or <Sue>, or any member of the <Sales2> role:

```
GRANT SET USER (Joe, Sue), (ANY WITH ROLES Sales2) TO Sarah
```

This statement grants `<Joan>` the ability to impersonate any member of the `<Marketing1>` or `<Marketing2>` roles:

```
GRANT SET USER (ANY WITH ROLES Marketing1, Marketing2) TO Joan
```

Related Information

GRANT SET USER Privilege Statement

3.6.3 Starting to Impersonate Another User

Allows a user to impersonate another user by assuming that user's exact roles and system privileges. Impersonation remains active until it is explicitly stopped or the current session ends.

Prerequisites

The impersonator and target users meet all the requirements for impersonation. See Requirements for Impersonation [page 38].

Context

At-least criteria is validated when the SETUSER statement is executed, not when the SET USER system privilege is granted. When the SETUSER statement is executed, if the impersonating user fails to meet all at-least criteria, a `permission denied` message appears, and impersonation does not begin. However, if all at-least criteria is met, impersonation begins.

Once you issue the SETUSER statement, and impersonation begins, it remains in effect until:

- you manually terminated the impersonation
- you begin impersonating another user
- the current session ends

While a user is impersonating another user, roles and privileges and their related administrative rights can be granted or revoked from the impersonator or impersonatee as long as doing so does not violate the at-least criteria behind the impersonation. If the grant or revoke violates the criteria, an error message appears, and the grant or revoke statement fails. SAP recommends that impersonation be terminated as soon as the required tasks are complete.

Procedure

At a command prompt, type:

```
SETUSER <userID>
```

Related Information

SETUSER Statement

3.6.4 Verifying the Current Impersonation Status of a User

A successful impersonation remains in effect until it is manually terminated or the session is terminated.

To verify the current status of an impersonation, execute the following statement on the connection where the SETUSERstatement was issued:

```
SELECT CURRENT USER
```

This statement returns the name of the user the connection recognizes as the currently logged in user. If it is the expected user for the machine, no impersonation is active on the connection. If a different user name appears, it represents the user currently being impersonated on the connection.

Example

This example returns the name of the user of the current connection:

```
SELECT CURRENT USER
```

Current User
USER1

Activate impersonation of User2:

```
SETUSER USER2
```

Verify impersonation:

```
SELECT CURRENT USER
```

Current User
USER2

Terminate impersonation:

```
SETUSER
```

Verify impersonation:

```
SELECT CURRENT USER
```

Current User
USER1

3.6.5 Stopping Impersonation of Another User

End the impersonation of another user on the current connection. Once begun, impersonation of another user remains in effect until impersonation is stopped, or the current session ends.

Prerequisites

- The SETUSER statement is issued from the same connection where it was initiated.

Procedure

At a command prompt, execute:

```
SETUSER
```

Related Information

SETUSER Statement

3.6.6 Revoking the SET USER System Privilege from a User

Remove the ability of a user to impersonate other users, and to administer the SET USER system privilege.

Prerequisites

- Requires the SET USER system privilege granted with administrative rights.

Context

The SET USER system privilege can be granted to a user multiple times, using different clauses. For example, User1 is granted the SET USER system privilege once using the ANY clause and again with the<target_users_list> clause. In cases of multiple grants, the same form of the clause used for the grant must be used to revoke it. If the system privilege is revoked from User1 using the ANY clause, the grant with the <target_users_list> clause remains in effect. The net effect is that User1 is now limited to impersonating users on the <target_users_list>. Alternately, if the system privilege is revoked from User1 using the <target_users_list> clause, the grant with the ANY clause remains in effect. The net effect in this scenario is that User1 can continue to impersonate any user in the database.

> **Note**
>
> These examples assume User1 meets the at-least criteria for successful impersonation.

Procedure

To revoke the SET USER system privilege, execute one of these statements:

Revoke Type	Description
Administrative rights to system privilege only	`REVOKE ADMIN OPTION FOR SET USER ( ANY )` `FROM <user_ID [,...]>`
System privilege to impersonate any database user, including administrative rights	`REVOKE SET USER FROM <user_ID [,...]>`
System privilege to impersonate specified users	`REVOKE SET USER ( <target_users_list> )` `FROM <user_ID [,...]>`
System privilege to impersonate specified roles	`REVOKE SET USER ( ANY WITH ROLES <target_roles_list> )` `FROM <user_ID [,...]>`

Example

These statements remove the ability for `User1` to impersonate any database user:

```
REVOKE SET USER (ANY) FROM user1;
or
REVOKE SET USER FROM user1;
```

This statement removes administrative rights only to the `SET USER` system privilege from `user2`. `user2` can still impersonate any user in the database.

```
REVOKE ADMIN OPTION FOR SET USER (ANY) FROM user2;
```

This statement removes the ability of `user3` and `user4` to impersonate `user1`, `user2`, or `user3`.

```
REVOKE SET USER (user1, user2, user3) FROM user3, user4;
```

This statement removes the ability of `user3` to impersonate any member of the `<Sales1>` role:

```
REVOKE SET USER (ANY WITH ROLES Sales1) FROM user3;
```

This statement removes the ability of `user1` to impersonate `user2`, `user3`, or any member of the `<Sales2>` role:

```
REVOKE SET USER (user2, user3), (ANY WITH ROLES Sales2) FROM user1;
```

This statement removes the ability of `user1` to impersonate any member of the `Marketing1` or `Marketing2` roles:

```
REVOKE SET USER (ANY WITH ROLES Marketing1, Markeing2) FROM user1;
```

Related Information

REVOKE SET USER Privilege Statement

4 Authorization

4.1 Privileges

A privilege grants users the ability to perform an authorized operation on the system. For example, altering a table is a privileged operation, depending on the type of alteration you are making.

There are two types of privileges: system privileges and object privileges.

System privileges give you the general right to perform a privileged operation, while object privileges restrict you to performing the operation on a specific object. For example, if you have the `ALTER ANY TABLE` system privilege, you can alter any table in the database. If you have the `ALTER` object privilege, then you can alter only the specific objects on which that privilege has been granted to you.

System privileges are built in to the database and can be granted or revoked, but not created or dropped. With the exception of the MANAGE ROLES and UPGRADE ROLE privileges, system privileges cannot be modified. Each system privilege, with the exception of the SET USER system privilege, is granted by default to either the SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE role, but not both. The SET USER system privilege is granted to both roles.

See System Privileges [page 49] and Object-Level Privileges [page 72].

In this section:

4.1.1 Privileges Versus Permissions

In role-based security, privilege and permission are distinct concepts. A user may hold the privilege required for an authorized task, yet still lack permission to perform that task on a specific object or in a specific context.

A privilege grants a user or role the ability to perform a defined operation. Permission reflects whether that operation is allowed for the target object and in the current execution context.

Before running a privileged operation, the system validates privilege at three levels:

- Privileged operation
- Privileged operation on the acted-on object
- Privileged operation in the context in which he or she is attempting it

If privilege is missing at any level, permission is denied, the operation fails, and an error message is returned.

4.1.2 System Privileges

System privileges let you control access to authorized system operations. Each privileged database task on the server requires specific system privileges. System privileges can be granted individually to users or roles.

When a system privilege is granted to a role, all members of the role inherit the system privilege. All new members of a role automatically inherit all of the underlying system privileges of a role.

Each system privilege, with the exception of the SET USER system privilege, by default, is granted to either the SYS_AUTH_SA_ROLE or the SYS_AUTH_SSO_ROLE role, but not both. The exception, SET USER system privilege, is granted to both roles.

Individually granting the underlying system privileges of a role is semantically equivalent to granting the role itself. You can grant system privileges to multiple user-defined system roles in any combination to meet the functional security requirements of your organization.

With the exception of MANAGE ROLES and UPGRADE ROLE, you cannot modify system privileges. System privileges can be granted to, and revoked from, roles and users, but they cannot be dropped. System privileges cannot own objects.

In this section:

System Privileges Introduced in Upgrades [page 70]
A new release of the software may introduce new system privileges.

4.1.2.1 Alphabetical List of System Privileges

A list of all system privileges.

System privileges control the rights of users to perform authorized database tasks.

The following is a list of available system privileges:

System Privilege	Allows a User To...	Functional Area
(Deprecated) CREATE EXTERNAL REFERENCE	• Allows a user to create external references in the database. • You must have the system privileges required to create specific database objects before you can create external references. • For example, creating a self-owned text configuration object that uses an external term breaker requires both the CREATE TEXT CONFIGURATION and CREATE EXTERNAL REFERENCE system privileges.	External Environment
(Deprecated) MANAGE ANY EXTERNAL ENVIRONMENT	Allows a user to alter, comment on, start, and stop external environments.	External Environment
(Deprecated) MANAGE ANY EXTERNAL OBJECT	Allows a user to install, comment on, and remove external environment objects.	External Environment
ACCESS SERVER LS	Allows logical server connection using the SERVER logical server context.	Multiplex
ACCESS USER PASSWORD	Allows a user to access views that contain password hashes, and perform operations that involve accessing passwords, such as unloading, extracting, or comparing database	User and Login Management
ALTER ANY INDEX	Allows a user to alter and comment on indexes and text indexes on tables and views owned by any user.	Indexes
ALTER ANY MATERIALIZED VIEW	Allows a user to alter and comment on materialized views owned by any user.	Materialized Views

System Privilege	Allows a User To...	Functional Area
ALTER ANY OBJECT	Allows a user to alter and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	Objects
ALTER ANY OBJECT OWNER	Allows a user to alter the owner of any type of table object. This privilege does not allow changing of the owner of other objects, such as procedures, materialized views, and so on.	Objects
ALTER ANY PROCEDURE	Allows a user to alter and comment on procedures and functions owned by any user.	Procedures
ALTER ANY SEQUENCE	Allows a user to alter sequence generators owned by any user.	Sequence
ALTER ANY TABLE	Allows a user to: • Alter and comment on tables (including proxy tables) owned by any user. • Truncate tables, table partitions, or views owned by any user • Comment on tables (including proxy tables) and columns in tables owned by any user.	Tables
ALTER ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.	Text Configuration
ALTER ANY TRIGGER	Allows a user to: • Alter triggers on tables and views. • Issue comments on tables (also requires the ALTER object-level privilege on the table).	Triggers
ALTER ANY VIEW	Allows a user to alter and comment on views owned by any user.	Views

System Privilege	Allows a User To...	Functional Area
ALTER DATABASE	Allows a user to: • Upgrade a database. • Perform cost model calibration. • Load database statistics. • Alter transaction logs (also requires the SERVER OPERATOR system privilege). • Change ownership of the database (also requires the MANAGE ANY MIRROR SERVER system privilege).	Database
ALTER DATATYPE	Allows a user to alter data types.	Data Type
BACKUP ANY TABLE	Allows a user to back up a table.	Database
BACKUP DATABASE	Allows a user to back up a database.	Database
BACKUP OWNER TABLE	Allows a user to back up self-owned tables.	Database
CHANGE PASSWORD	• Allows a user to manage user passwords for any user. • This system privilege can apply to all users, or it can be limited to a set of specified users, or users who are granted one or more specified roles. • This system privilege is not required to change a user's own password.	User and Login Management
CHECKPOINT	Allows a user to force the database server to execute a checkpoint.	Database
COMMENT ANY OBJECT	Allows a user to comment on any type of object owned by any user that can be created using the CREATE ANY OBJECT system privilege.	Objects
CREATE ANY INDEX	Allows a user to create and comment on indexes and text indexes on tables and views owned by any user.	Indexes
CREATE ANY MATERIALIZED VIEW	Allows a user to create and comment on materialized views owned by any user.	Materialized Views
CREATE ANY MUTEX SEMAPHORE	Allows a user to create a mutex or semaphore owned by any user.	Mutex and Semaphores

System Privilege	Allows a User To...	Functional Area
CREATE ANY OBJECT	Allows a user to create and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	Objects
CREATE ANY PROCEDURE	Allows a user to create and comment on procedures and functions owned by any user.	Procedure
CREATE ANY SEQUENCE	Allows a user to create sequence generators, regardless of owner.	Sequence
CREATE ANY TABLE	Allows a user to: • Create and comment on tables (including proxy tables) owned by any user. • Comment on columns in tables (including proxy tables) owned by any user.	Table
CREATE ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.	Text Configuration
CREATE ANY TRIGGER	Allows a user to create and comment (also requires the ALTER object level privilege on the table) on tables and views.	Triggers
CREATE ANY VIEW	Allows a user to create and comment on views owned by any user.	Views
CREATE DATABASE VARIABLE	Allows a user to create, select from, update, and drop self-owned database-scope variables.	Database Variables
CREATE DATATYPE	Allows a user to create data types.	Database
CREATE MATERIALIZED VIEW	Allows a user to create and comment on self-owned materialized views.	Materialized Views
CREATE MESSAGE	Allows a user to create messages.	Miscellaneous
CREATE PROCEDURE	Allows a user to create and comment on self-owned procedures and functions. create a self-owned stored procedure or function.	Procedure

System Privilege	Allows a User To...	Functional Area
CREATE PROXY TABLE	Allows a user to create self-owned proxy tables.	Table
CREATE TABLE	Allows a user to: • Create self-owned tables. • Comment on self-owned columns and tables.	Table
CREATE TEXT CONFIGURATION	Allows a user to create and comment on self-owned text configuration objects.	Text Configuration
CREATE VIEW	Allows a user to create and comment on self-owned views. Required to create self-owned views.	Views
DEBUG ANY PROCEDURE	Allows a user to debug any database object.	Miscellaneous
DELETE ANY TABLE	Allows a user to delete rows in tables and views owned by any user.	Table
DROP ANY INDEX	Allows a user to drop indexes and text indexes on tables and views owned by any user.	Indexes
DROP ANY MATERIALIZED VIEW	Allows a user to drop materialized views owned by any user.	Materialized View
DROP ANY MUTEX SEMAPHORE	Allows a user to drop a mutex or semaphore owned by any user.	Mutex and Semaphores
DROP ANY OBJECT	Allows a user to drop the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views	Objects
DROP ANY PROCEDURE	Allows a user to drop procedures and functions owned by any user.	Procedure
DROP ANY SEQUENCE	Allows a user to drop sequence generators owned by any user.	Sequence
DROP ANY TABLE	Allows a user to drop tables (including proxy tables) owned by any user.	Table

System Privilege	Allows a User To...	Functional Area
DROP ANY TEXT CONFIGURATION	Allows a user to drop text configuration objects owned by any user.	Text Configuration
DROP ANY VIEW	Allows a user to drop views owned by any user.	Views
DROP CONNECTION	Allows a user to drop any connections to the database.	Database
DROP DATATYPE	Allows a user to drop data types.	Database
DROP MESSAGE	Allows a user to drop messages.	Miscellaneous
EXECUTE ANY PROCEDURE	Allows a user to execute procedures and functions owned by any user.	Procedure
INSERT ANY TABLE	Allows a user to insert rows into tables and views owned by any user.	Table
LOAD ANY TABLE	Allows a user to load data into tables owned by any user.	Table
MANAGE ANY DATABASE VARIABLE	Allows a user to create and drop database-scope variables owned by self or by PUBLIC.	Database Variables
MANAGE ANY DBSPACE	Allows a user to: • Create, alter, drop, and comment on dbspaces. • Grant and revoke CREATE object-level privileges on dbspaces. • Move data to any dbspace. • Issue a read-only selective restore statement on any dbspace. • Run the database delete file function.	Dbspaces
MANAGE ANY EVENT	Allows a user to create, alter, drop, trigger, and comment on events.	Miscellaneous
MANAGE ANY LDAP SERVER	Allows a user to create, alter, drop, validate, and comment on LDAP servers.	Miscellaneous
MANAGE ANY LOGIN POLICY	Allows a user to create, alter, drop, and comment on login policies.	User and Login Management
MANAGE ANY MIRROR SERVER	Allows a user to: • Create, alter, drop, and comment on mirror servers. • Change mirror server parameters. • Set mirror server options. • Change ownership of a database.	Miscellaneous

System Privilege	Allows a User To...	Functional Area
MANAGE ANY OBJECT PRIVILEGE	Allows a user to: • Grant and revoke SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES, LOAD, and TRUNCATE object-level privileges on tables owned by any user. • Grant and revoke SELECT, INSERT, DELETE, and UPDATE object-level privileges on views owned by any user. • Grant and revoke EXECUTE privileges on procedures and functions owned by any user. • Grant and revoke USAGE object-level privileges on sequence generators owned by any user. • Grant and revoke CREATE object-level privileges on dbspaces.	Objects
MANAGE ANY PROPERTY HISTORY	Allows a user to turn on and configure the tracking of database server property values.	Server Operator
MANAGE ANY SPATIAL OBJECT	Allows a user to create, alter, drop, and comment on spatial reference systems and spatial unit of measures.	Miscellaneous
MANAGE ANY STATISTICS	Allows a user to create, alter, drop, and update database statistics for any table.	Miscellaneous
MANAGE ANY USER	Allows a user to: • Create, alter, drop, and comment on database users (including assigning an initial password). • Force a password change on next login for any user. • Assign and reset the login policy for any user. • Create, drop, and comment on integrated logins and Kerberos logins. • Create and drop external logins.	User and Login Management
MANAGE ANY WEB SERVICE	Allows a user to create, alter, drop, and comment on web services.	Miscellaneous
MANAGE AUDITING	Allows a user to run the sa_audit_string stored procedure.	Procedure
MANAGE CCERTIFICATES	Allows a user to create, alter, drop, and comment on certificates.	Miscellaneous
MANAGE LISTENERS	Allows a user to start and stop network listeners.	Server Operator
MANAGE MULTIPLEX	Allows users to: • Create, alter, drop, or comment on logical servers and logical server policies. • Assign a dbspace to logical servers. • Release a populated dbspace from the exclusive use of a logical server. • Manages failover configurations, and perform a manual failover.	Multiplex
MANAGE PROFILING	Allows a user to manage database server tracing. The DIAGNOSTICS system role is also required to fully utilize diagnostics functionality for user information.	Database

System Privilege	Allows a User To...	Functional Area
MANAGE ROLES	• Allows a user to create new roles and act as a global administrator for new and existing roles. By default, MANAGE ROLES is granted administrative rights on each newly created role. A user requires administrative rights on the role to delete it. • Administration of a role can also be granted directly to users either during or after the creation of the role. When granted directly to a user, the user does not require the MANAGE ROLES system privilege to administer the role. • If no role administrator is specified during role creation, the MANAGE ROLES system privilege is automatically granted to the role with the ADMIN ONLY OPTION clause, allowing the global role administrator to administer the role. If at least one role administrator is specified during creation, the MANAGE ROLES system privilege is not granted to the role, and global role administrators cannot manage the role.	Roles
MONITOR	Allows a user to monitor a database, including accessing privileged statistics, connected users, and locks.	Database
OFLINE RESET PASSWORD	Allows the user to reset another user's password when using the database server -orp option. The user to perform the reset should not be the DBA user.	User and Login Management
READ CLIENT FILE	Allows a user to read files on the client computer.	Files
READ FILE	Allows a user to read files on the database server computer.	Files
REORGANIZE ANY OBJECT	Allows a user to reorganize tables and materialized views owned by any user.	Objects
RESTORE ANY TABLE	Allows a user to restore any table.	Database
RESTORE OWNER TABLE	Allows a user to restore self-owned tables.	Database
SELECT ANY TABLE	Allows a user to query tables and views owned by any user.	Table
SELECT PUBLIC DATABASE VARIABLE	Allows a user to select the value of a database-scope variable owned by PUBLIC.	Database Variables
SERVER OPERATOR	Allows a user to: • Create, drop, change ownership of a database, and restore the catalog (only). • Create, alter, and drop a server. • Manage a server cache. • Start and stop database or database engine. • Encrypt databases. • Change a database transaction log.	Server Operator
SET ANY PUBLIC OPTION	Allows a user to set PUBLIC database options that do not require the SET ANY SECURITY OPTION or the SET ANY SYSTEM OPTION system privileges.	Database Options
SET ANY SECURITY OPTION	Allows a user to set any PUBLIC security database options.	Database Options

System Privilege	Allows a User To...	Functional Area
SET ANY SYSTEM OPTION	Allows a user to set PUBLIC system database options.	Database Options
SET ANY USER DEFINED OPTION	Allows a user to set user-defined database options.	Database Options
SET USER (granted with administrative rights only)	• Allows a user to temporarily assume the roles and privileges of another user. • This system privilege can apply to all users, or can be limited to a set of specified users, or users who are granted one or more specified roles.	User and Login Management
TRUNCATE ANY TABLE	Allows a user to truncate data for tables and materialized views owned by any user.	Table
UPDATE ANY MUTEX SEMAPHORE	Allows a user to update a mutex or semaphore owned by any user.	Mutex and Semaphores
UPDATE ANY TABLE	Allows a user to update rows in tables and views owned by any user.	Table
UPDATE PUBLIC DATABASE VARIABLE	Allows a user to update database-scope variables owned by PUBLIC.	Database Variables
UPGRADE ROLE	Allows a user to be a default administrator of any system privilege that is introduced when upgrading an SAP IQ database from version 16.0. By default, the UPGRADE ROLE system privilege is granted to the SYS_AUTH_SA_ROLE role, if it exists.	Roles
USE ANY SEQUENCE	Allows a user to use sequence generators owned by any user.	Sequence
VALIDATE ANY OBJECT	Allows a user to validate tables, materialized views, indexes, and text indexes owned by any user.	Objects
WRITE CLIENT FILE	Allows a user to write files to the client computer.	Files
WRITE FILE	Allows a user to write files on the database server computer.	Files

4.1.2.2 Functional Area List of System Privileges

A list of system privileges organized by functional area.

Functional Area	System Privilege	Allows a User To...
Data Type	ALTER DATATYPE	Allows a user to alter data types.

Functional Area	System Privilege	Allows a User To...
Database	ALTER DATABASE	Allows a user to: • Upgrade a database. • Perform cost model calibration. • Load database statistics. • Alter transaction logs (also requires the SERVER OPERATOR system privilege). • Change ownership of the database (also requires the MANAGE ANY MIRROR SERVER system privilege).
	BACKUP ANY TABLE	Allows a user to back up a table.
	BACKUP DATABASE	Allows a user to back up a database.
	BACKUP OWNER TABLE	Allows a user to back up self-owned tables.
	CHECKPOINT	Allows a user to force the database server to execute a checkpoint.
	CREATE DATATYPE	Allows a user to create data types.
	DROP CONNECTION	Allows a user to drop any connections to the database.
	DROP DATATYPE	Allows a user to drop data types.
	MANAGE PROFILING	Allows a user to manage database server tracing. The DIAGNOSTICS system role is also required to fully utilize diagnostics functionality for user information.
	MONITOR	Allows a user to monitor a database, including accessing privileged statistics, connected users, and locks.
	RESTORE ANY TABLE	Allows a user to restore any table.
	RESTORE OWNER TABLE	Allows a user to restore self-owned tables.
Database Options	SET ANY PUBLIC OPTION	Allows a user to set PUBLIC database options that do not require the SET ANY SECURITY OPTION or the SET ANY SYSTEM OPTION system privileges.
	SET ANY SECURITY OPTION	Allows a user to set any PUBLIC security database options.
	SET ANY SYSTEM OPTION	Allows a user to set PUBLIC system database options.
	SET ANY USER DEFINED OPTION	Allows a user to set user-defined database options.
Database Variables	CREATE DATABASE VARIABLE	Allows a user to create, select from, update, and drop self-owned database-scope variables.
	MANAGE ANY DATABASE VARIABLE	Allows a user to create and drop database-scope variables owned by self or by PUBLIC.
	SELECT PUBLIC DATABASE VARIABLE	Allows a user to select the value of a database-scope variable owned by PUBLIC.
	UPDATE PUBLIC DATABASE VARIABLE	Allows a user to update database-scope variables owned by PUBLIC.

Functional Area	System Privilege	Allows a User To...
Dbspaces	MANAGE ANY DBSPACE	Allows a user to: • Create, alter, drop, and comment on dbspaces. • Grant and revoke CREATE object-level privileges on dbspaces. • Move data to any dbspace. • Issue a read-only selective restore statement on any dbspace. • Run the database delete file function.
External Environment	(Deprecated) CREATE EXTERNAL REFERENCE	• Allows a user to create external references in the database. • You must have the system privileges required to create specific database objects before you can create external references. • For example, creating a self-owned text configuration object that uses an external term breaker requires both the CREATE TEXT CONFIGURATION and CREATE EXTERNAL REFERENCE system privileges.
	(Deprecated) MANAGE ANY EXTERNAL ENVIRONMENT	Allows a user to alter, comment on, start, and stop external environments.
	(Deprecated) MANAGE ANY EXTERNAL OBJECT	Allows a user to install, comment on, and remove external environment objects.
Files	READ CLIENT FILE	Allows a user to read files on the client computer.
	READ FILE	Allows a user to read files on the database server computer.
	WRITE CLIENT FILE	Allows a user to write files to the client computer.
	WRITE FILE	Allows a user to write files on the database server computer.
Indexes	ALTER ANY INDEX	Allows a user to alter and comment on indexes and text indexes on tables and views owned by any user.
	CREATE ANY INDEX	Allows a user to create and comment on indexes and text indexes on tables and views owned by any user.
	DROP ANY INDEX	Allows a user to drop indexes and text indexes on tables and views owned by any user.
Materialized View	DROP ANY MATERIALIZED VIEW	Allows a user to drop materialized views owned by any user.
Materialized Views	ALTER ANY MATERIALIZED VIEW	Allows a user to alter and comment on materialized views owned by any user.
	CREATE ANY MATERIALIZED VIEW	Allows a user to create and comment on materialized views owned by any user.
	CREATE MATERIALIZED VIEW	Allows a user to create and comment on self-owned materialized views.
Miscellaneous	CREATE MESSAGE	Allows a user to create messages.
	DEBUG ANY PROCEDURE	Allows a user to debug any database object.
	DROP MESSAGE	Allows a user to drop messages.

Functional Area	System Privilege	Allows a User To...
	MANAGE ANY EVENT	Allows a user to create, alter, drop, trigger, and comment on events.
	MANAGE ANY LDAP SERVER	Allows a user to create, alter, drop, validate, and comment on LDAP servers.
	MANAGE ANY MIRROR SERVER	Allows a user to: • Create, alter, drop, and comment on mirror servers. • Change mirror server parameters. • Set mirror server options. • Change ownership of a database.
	MANAGE ANY SPATIAL OBJECT	Allows a user to create, alter, drop, and comment on spatial reference systems and spatial unit of measures.
	MANAGE ANY STATISTICS	Allows a user to create, alter, drop, and update database statistics for any table.
	MANAGE ANY WEB SERVICE	Allows a user to create, alter, drop, and comment on web services.
Multiplex	ACCESS SERVER LS	Allows logical server connection using the SERVER logical server context.
	MANAGE MULTIPLEX	Allows users to: • Create, alter, drop, or comment on logical servers and logical server policies. • Assign a dbspace to logical servers. • Release a populated dbspace from the exclusive use of a logical server. • Manages failover configurations, and perform a manual failover.
Mutex and Semaphores	CREATE ANY MUTEX SEMAPHORE	Allows a user to create a mutex or semaphore owned by any user.
	DROP ANY MUTEX SEMAPHORE	Allows a user to drop a mutex or semaphore owned by any user.
	UPDATE ANY MUTEX SEMAPHORE	Allows a user to update a mutex or semaphore owned by any user.

Functional Area	System Privilege	Allows a User To...
Objects	ALTER ANY OBJECT	Allows a user to alter and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views
	ALTER ANY OBJECT OWNER	Allows a user to alter the owner of any type of table object. This privilege does not allow changing of the owner of other objects, such as procedures, materialized views, and so on.
	COMMENT ANY OBJECT	Allows a user to comment on any type of object owned by any user that can be created using the CREATE ANY OBJECT system privilege.
	CREATE ANY OBJECT	Allows a user to create and comment on the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views

Functional Area	System Privilege	Allows a User To...
	DROP ANY OBJECT	Allows a user to drop the following types of objects owned by any user: • Data types • Events • Functions • Indexes • Materialized views • Messages • Procedures • Sequence generators • Spatial reference systems • Spatial units of measure • Statistics • Tables • Text configuration objects • Text indexes • Triggers • Views
	MANAGE ANY OBJECT PRIVILEGE	Allows a user to: • Grant and revoke SELECT, INSERT, DELETE, UPDATE, ALTER, REFERENCES, LOAD, and TRUNCATE object-level privileges on tables owned by any user. • Grant and revoke SELECT, INSERT, DELETE, and UPDATE object-level privileges on views owned by any user. • Grant and revoke EXECUTE privileges on procedures and functions owned by any user. • Grant and revoke USAGE object-level privileges on sequence generators owned by any user. • Grant and revoke CREATE object-level privileges on dbspaces.
	REORGANIZE ANY OBJECT	Allows a user to reorganize tables and materialized views owned by any user.
	VALIDATE ANY OBJECT	Allows a user to validate tables, materialized views, indexes, and text indexes owned by any user.
Procedure	CREATE ANY PROCEDURE	Allows a user to create and comment on procedures and functions owned by any user.
	CREATE PROCEDURE	Allows a user to create and comment on self-owned procedures and functions. create a self-owned stored procedure or function.
	DROP ANY PROCEDURE	Allows a user to drop procedures and functions owned by any user.
	EXECUTE ANY PROCEDURE	Allows a user to execute procedures and functions owned by any user.

Functional Area	System Privilege	Allows a User To...
	MANAGE AUDITING	Allows a user to run the sa_audit_string stored procedure.
Procedures	ALTER ANY PROCEDURE	Allows a user to alter and comment on procedures and functions owned by any user.
Roles	MANAGE ROLES	• Allows a user to create new roles and act as a global administrator for new and existing roles. By default, MANAGE ROLES is granted administrative rights on each newly created role. A user requires administrative rights on the role to delete it. • Administration of a role can also be granted directly to users either during or after the creation of the role. When granted directly to a user, the user does not require the MANAGE ROLES system privilege to administer the role. • If no role administrator is specified during role creation, the MANAGE ROLES system privilege is automatically granted to the role with the ADMIN ONLY OPTION clause, allowing the global role administrator to administer the role. If at least one role administrator is specified during creation, the MANAGE ROLES system privilege is not granted to the role, and global role administrators cannot manage the role.
	UPGRADE ROLE	Allows a user to be a default administrator of any system privilege that is introduced when upgrading an SAP IQ database from version 16.0. By default, the UPGRADE ROLE system privilege is granted to the SYS_AUTH_SA_ROLE role, if it exists.
Sequence	ALTER ANY SEQUENCE	Allows a user to alter sequence generators owned by any user.
	CREATE ANY SEQUENCE	Allows a user to create sequence generators, regardless of owner.
	DROP ANY SEQUENCE	Allows a user to drop sequence generators owned by any user.
	USE ANY SEQUENCE	Allows a user to use sequence generators owned by any user.
Server Operator	MANAGE ANY PROPERTY HISTORY	Allows a user to turn on and configure the tracking of database server property values.
	MANAGE LISTENERS	Allows a user to start and stop network listeners.
	SERVER OPERATOR	Allows a user to: • Create, drop, change ownership of a database, and restore the catalog (only). • Create, alter, and drop a server. • Manage a server cache. • Start and stop database or database engine. • Encrypt databases. • Change a database transaction log.

Functional Area	System Privilege	Allows a User To...
Table	CREATE ANY TABLE	Allows a user to: • Create and comment on tables (including proxy tables) owned by any user. • Comment on columns in tables (including proxy tables) owned by any user.
	CREATE PROXY TABLE	Allows a user to create self-owned proxy tables.
	CREATE TABLE	Allows a user to: • Create self-owned tables. • Comment on self-owned columns and tables.
	DELETE ANY TABLE	Allows a user to delete rows in tables and views owned by any user.
	DROP ANY TABLE	Allows a user to drop tables (including proxy tables) owned by any user.
	INSERT ANY TABLE	Allows a user to insert rows into tables and views owned by any user.
	LOAD ANY TABLE	Allows a user to load data into tables owned by any user.
	SELECT ANY TABLE	Allows a user to query tables and views owned by any user.
	TRUNCATE ANY TABLE	Allows a user to truncate data for tables and materialized views owned by any user.
	UPDATE ANY TABLE	Allows a user to update rows in tables and views owned by any user.
Tables	ALTER ANY TABLE	Allows a user to: • Alter and comment on tables (including proxy tables) owned by any user. • Truncate tables, table partitions, or views owned by any user • Comment on tables (including proxy tables) and columns in tables owned by any user.
Text Configuration	ALTER ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.
	CREATE ANY TEXT CONFIGURATION	Allows a user to alter and comment on text configuration objects owned by any user.
	CREATE TEXT CONFIGURATION	Allows a user to create and comment on self-owned text configuration objects.
	DROP ANY TEXT CONFIGURATION	Allows a user to drop text configuration objects owned by any user.
Triggers	ALTER ANY TRIGGER	Allows a user to: • Alter triggers on tables and views. • Issue comments on tables (also requires the ALTER object-level privilege on the table).

Functional Area	System Privilege	Allows a User To...
	CREATE ANY TRIGGER	Allows a user to create and comment (also requires the ALTER object level privilege on the table) on tables and views.
User and Login Management	ACCESS USER PASSWORD	Allows a user to access views that contain password hashes, and perform operations that involve accessing passwords, such as unloading, extracting, or comparing database
	CHANGE PASSWORD	• Allows a user to manage user passwords for any user. • This system privilege can apply to all users, or it can be limited to a set of specified users, or users who are granted one or more specified roles. • This system privilege is not required to change a user's own password.
	MANAGE ANY LOGIN POLICY	Allows a user to create, alter, drop, and comment on login policies.
	MANAGE ANY USER	Allows a user to: • Create, alter, drop, and comment on database users (including assigning an initial password). • Force a password change on next login for any user. • Assign and reset the login policy for any user. • Create, drop, and comment on integrated logins and Kerberos logins. • Create and drop external logins.
	SET USER (granted with administrative rights only)	• Allows a user to temporarily assume the roles and privileges of another user. • This system privilege can apply to all users, or can be limited to a set of specified users, or users who are granted one or more specified roles.
Views	ALTER ANY VIEW	Allows a user to alter and comment on views owned by any user.
	CREATE ANY VIEW	Allows a user to create and comment on views owned by any user.
	CREATE VIEW	Allows a user to create and comment on self-owned views. Required to create self-owned views.
	DROP ANY VIEW	Allows a user to drop views owned by any user.

4.1.2.3 Granting and Revoking a System Privilege from User and Roles

Allow granting or revoking of specific system privileges to specific users, with or without administrative rights.

Prerequisites

- Requires administrative privilege over the system privilege being granted or revoked.
- If the grant or revoke is to a role, also requires administrative privilege over the role.

Context

If not specified, the default grant is `WITH NO ADMIN OPTION`.

The syntax to grant the CHANGE PASSWORD and SET USER system privileges uses different syntax from the basic syntax. See GRANT SET USER Privilege Statement and GRANT CHANGE PASSWORD Privilege Statement.

> ⚠ Caution
>
> When regranting a system privilege, if the grantee already has the system privilege granted with rights higher than the current grant, the grantee retains the original rights and inherits the higher rights. But if the regrant is for lesser rights than the original grant, the higher rights are `NOT` reduced for the grantee.

Procedure

To grantor revoke a system privilege to a user, execute one of these statements:

Grant Type	Statement

Grant system privilege with no administrative rights.	`GRANT <system_privilege> TO <grantee>[,...]`
	Example:
	`GRANT CREATE TABLE TO USER1, ROLE_1;`

Grant system privilege with administrative rights only.	`GRANT <system_privilege> TO <grantee>[,...]` `WITH ADMIN ONLY OPTION`

Grant Type	Statement

	Example:
	`GRANT CREATE TABLE TO USER1, ROLE_1`
	`  WITH ADMIN ONLY OPTION;`

Grant system privilege with full administrative rights.	`GRANT <system_privilege> TO <grantee>[,...]`
	`  WITH ADMIN OPTION`
	Example:
	`GRANT CREATE TABLE TO USER1, ROLE_1`
	`  WITH ADMIN OPTION;`

Revoke administrative rights only on the system privilege.	`REVOKE ADMIN OPTION FOR <system_privilege>`
	`FROM <grantee>[,...]`
	Example:
	`REVOKE ADMIN OPTION FOR CREATE TABLE FROM`
	`USER1, ROLE_1;`

Revoke system privilege and administrative rights.	`REVOKE <system_privilege> FROM`
	`<grantee>[,...]`
	Example:
	`REVOKE CREATE TABLE FROM USER1, ROLE_1;`

Related Information

GRANT System Privilege Statement
REVOKE System Privilege Statement

4.1.2.4 Listing System Privileges Granted to a User or Role

Display the roles and system privileges granted directly and indirectly to a user or role.

To view roles and system privileges granted to a user, see Listing Roles and System Privileges Granted to a User [page 28].

To view roles and system privileges granted to a role, see Listing the System Privileges and Roles Granted to a Role [page 126].

4.1.2.5 Determining If a System Privilege Is Granted to the Connected User

Determine whether a specific system privilege or role is granted to the invoking (connected) user.

Prerequisites

Requires one of:

- No additional privilege required.

Context

The result only indicates the status of the grant. It does not indicate the administrative rights level of the grant.

1. To determine if a system privilege is granted to the connected user, execute:

   ```
   SELECT sp_has_role ('<system-privilege-name>')
   ```

2. To determine if a role is granted to the connected user, execute:

   ```
   SELECT sp_has_role ('<role-name>')
   ```

Output:

Result	Means
1	The system privilege or role exists and is granted to the connected user.
0	The system privilege or role exists and is not granted to the connected user.
-1	The system privilege or role specified does not exist.

Examples

This examples check is the `ALTER ANY INDEX` system privilege is granted to the connected user.

```
SELECT sp_has_role ('ALTER ANY INDEX')
```

sp_has_role('ALTER ANY INDEX')
0

A result of 0 indicates the system privilege is not granted to the connected user.

This examples check is the `ALTER ANY INDEX` system privilege is granted to the connected user.

```
SELECT sp_has_role ('role1')
```

sp_has_role('role1')
1

A result of 1 indicates the connected user is a member of `role1`.

This examples check is the `role2` system privilege is granted to the connected user.

```
SELECT sp_has_role ('role2')
```

sp_has_role('role2')
-1

A result of -1 indicates that `role2` does not exist.

Related Information

sp_displayroles System Procedure

4.1.2.6 System Privileges Introduced in Upgrades

A new release of the software may introduce new system privileges.

In a *new database*, these privileges are automatically included in either the SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE compatibility roles, depending on the type of privileged operation they allow.

In an *upgraded database*, these privileges are only added to the UPGRADE ROLE system privilege . You then decide which roles and users you want to grant the new privileges to. After you have granted the new privileges, you should revoke them from the UPGRADE ROLE system privilege.

> ⚠ Caution
>
> Even though UPGRADE ROLE is a system privilege, it acts like a role because it can have system privileges granted to it.

In this section:

4.1.2.6.1 Distributing Privileges Granted to the UPGRADE ROLE System Privilege After an Upgrade

Grant the privileges that have been added to the UPGRADE ROLE system privilege to other users or roles (except for compatibility roles), and then revoke the privileges from UPGRADE ROLE.

Prerequisites

You must have exercise and administration rights for the UPGRADE ROLE system privilege.

Context

Perform this procedure after upgrading when new privileges have been added to the UPGRADE ROLE system privilege. To determine if new privileges have been added for the release, use the sp_displayroles procedure to view the roles and privileges for the UPGRADE ROLE. You can also look for mentions of newly added privileges in the list of new features for the release.

If you simply upgrade a database without following the steps described here, the database may encounter errors during a subsequent rebuild.

Procedure

1. In Interactive SQL, log in as a user with administration and exercise rights on the UPGRADE ROLE system privilege, and execute a statement that calls the sp_displayroles system procedure to display the privileges that have been granted to the UPGRADE ROLE system privilege. Note that when executing this statement, you must use the internal representation of the UPGRADE ROLE system privilege, which is SYS_UPGRADE_ROLE_ROLE:

   ```
   CALL sp_displayroles ( 'SYS_UPGRADE_ROLE_ROLE', 'expand_down' );
   ```

2. Grant the privileges listed to other users or roles, ensuring that you grant administration rights to at least one other user or role. You cannot assign the privilege to a compatibility role.
3. For each privilege you granted, log in as a user with administration rights for that privilege, and execute a REVOKE statement to revoke the privilege from the UPGRADE ROLE system privilege (again, use the internal representation, SYS_UPGRADE_ROLE_ROLE). For example:

   ```
   REVOKE <new-privilege-name> FROM SYS_UPGRADE_ROLE_ROLE;
   ```

Results

All new privileges have been granted to other users or roles, and the UPGRADE ROLE has no privileges granted to it.

Example

Here is an example for the OFFLINE RESET PASSWORD system privilege.

```
CALL sp_displayroles ( 'SYS_UPGRADE_ROLE_ROLE', 'expand_down' );
GRANT OFFLINE RESET PASSWORD TO DBA WITH ADMIN OPTION;
REVOKE OFFLINE RESET PASSWORD FROM SYS_UPGRADE_ROLE_ROLE;
```

Related Information

Listing the System Privileges and Roles Granted to a Role [page 126]
Granting and Revoking a System Privilege from User and Roles [page 67]

4.1.3 Object-Level Privileges

Object privileges control actions on database objects like tables, views, and stored procedures.

When granted, the privilege applies only to the specified object.

When a privilege is granted to a role, all members of the role inherit the privilege. All new members of a role automatically inherit the underlying privileges of a role.

Granting privileges to a role and then granting the role to a user is semantically equivalent to granting each underlying privilege directly to a user.

You can't modify or drop object privileges.

Ownership of a database object carries with it privileges to carry out actions on that object, but the creator of a database object may not necessarily be its owner. With sufficient privilege, another user can be designated as owner during the create process. If no owner is specified, the creator is the owner.

The owner of a table can modify the object, for example alter the table structure, or can grant privileges to other database users to query or update information within the table.

A user with the `ALTER ANY OBJECT` system privilege can modify any database object, regardless of owner. A user with the `CREATE ANY OBJECT` system privilege can create database objects to be owned by other users. A user with the `MANAGE ANY OBJECT` system privilege can grant object-level privileges on objects regardless of owner, but cannot use the object-level privileges on objects owned by other. These object-level privileges must be explicIty granted by the owner or a user granted administrative privilege on the object.

In this section:

4.1.3.1 Alphabetical Listing of Object Privileges

A list of all available object and schema privileges.

Object Privilege	Supported Object Types	Description
ALL	• All	• Grants ALTER, BACKUP TABLE, DELETE, INSERT, REFERENCES, RESTORE TABLE, SELECT, TRUNCATE, and UPDATE privileges on tables. • Grants DELETE, INSERT, LOAD and UPDATE privileges on views. • Grants ALTER, DELETE, SELECT, and TRUNCTATE on materialized views. • Grants REFERENCES on indexes. • Grants EXECUTE on procedures and functions. • Grants USAGE on sequences.
ALTER	• Tables • Materialized views	User can alter the named table or materialized view.

Object Privilege	Supported Object Types	Description
BACKUP TABLE	• Tables	User can backup the named table, regardless of table ownership. Additional BACKUP TABLE system privileges are required to backup a table. See BACKUP TABLE Statement.
DELETE	• Tables • Views	User can delete rows from the named table or view.
EXECUTE	• Procedures • Functions	User can execute the named procedure or function. EXECUTE does not support the WITH GRANT OPTION clause.
INSERT	• Tables • Views	User can insert rows into the named table or view.
LOAD	• Tables • Views	User can load data into the named table or view.
REFERENCES	• Tables	User can create indexes and foreign keys on the named table or view. If column names are specified, then the user can only see those columns. Columns can be specified only on a table, not an index or schema. For example, GRANT REFERENCE on T1(c1) means that you can create an index or foreign key on c1 of T1 only.
RESTORE TABLE	• Tables	User can restore the named table, regardless of table ownership. Additional RESTORE TABLE system privileges are required to restore a table. See RESTORE TABLE Statement
SELECT	• Tables • View • Materialized Views	User can look at information in a named table, view or materialized view. If column names are specified, then the user only sees information in the specified columns. Columns can be specified only on a table, not on a view, materialized view, or schema.
TRUNCATE	• Table • Materialized View	User can truncate the named table or materialized view.
UPDATE	• Tables • Views	User can update rows in the named view or table. If column names are specified, then the user can update only those columns. Columns can be named only on a table, not on a view or schema. To update a table, users must have both SELECT and UPDATE privilege on the table.

Object Privilege	Supported Object Types	Description
USAGE	• Sequences	User can evaluate the current or next value in a sequence.

In a multiplex, only write servers can modify table privileges on tables owned by the write server.

4.1.3.2 Granting and Revoking an Object Privilege

Grant or revoke object privileges on objects, such as tables, procedures, functions, or sequences. This privilege does not apply to views.

Prerequisites

Requires one of:

- You own the object.
- `MANAGE ANY OBJECT PRIVILEGE` system privilege
- Administrative rights on the specific object privilege being granted or revoked on the object.

To grant and revoke BACKUP and RESTORE TABLE object privileges requires additional privileges. See Granting and Revoking the RESTORE TABLE Privilege [page 78].

Context

Privileges can be granted on a specific object, with or without administrative rights. If not specified, no administrative rights are granted. When granted on a specific object, the privilege applies only to that object. You can specify multiple object privileges in a single statement, but they all apply to a single object.

Revoking privilege applies to the object privilege itself, not to any administrative right granted on the privilege. Therefore, you cannot revoke administrative rights only and leave the object privilege intact. To correctly remove a user's administrative rights only to an object privilege, first revoke the privilege and then regrant the privilege without the WITH GRANT OPTION clause.

If you revoke a privilege from a user who has been granted a privilege with the WITH GRANT OPTION clause, then everyone to whom that user granted the privilege also has his or her privilege revoked. For example, you granted User1 the SELECT privilege with the WITH GRANT OPTION clause. User1 then grants the SELECT privilege to User2. If you revoke the SELECT privilege from User1, it is also revoked from User2.

For a list of available privileges, see Alphabetical Listing of Object Privileges [page 73].

Procedure

To grantor revoke an object privilege to a user, execute one of these statements:

Grant Type	Statement
Grant object, schema, or PSE privilege with no administrative rights.	1. `GRANT <object-privilege>[,...] TO <object-name> TO <grantee>[,...]` 2. `GRANT <schema-privilege>[,...] TO SCHEMA <schema-name> TO <grantee>[,...]` 3. `GRANT <pse-privilege>[,...] TO PSE <pse-name> TO <grantee>[,...]` Example: 1. `GRANT ALTER, SELECT, UPDATE ON TABLE1 TO USER1, ROLE_1;` 2. `GRANT ALTER, SELCT ON SCHEMA MYSCHEMA1 TO USER1, ROLE_1;` 3. `GRANT ALTER, DROP ON PSE MYPSE1 TO USER1, ROLE_1;`
Grant the object, schema, or PSE privilege with full administrative rights.	1. `GRANT <object-privilege>[,...] TO <object-name> TO <grantee>[,...] [ WITH GRANT OPTION ]` 2. `GRANT <schema-privilege>[,...] TO SCHEMA <schema-name> TO <grantee>[,...] [ WITH GRANT OPTION ]` 3. `GRANT <pse-privilege>[,...] TO PSE <pse-name> TO <grantee>[,...] [ WITH GRANT OPTION ]` Example: 1. `GRANT SELECT, UPDATE ON TABLE1 TO USER1, ROLE_1 WITH GRANT OPTION;` 2. `GRANT SELECT, ALTER ON SCHEMA MYSCHEMA1 TO USER1, ROLE_1 WITH GRANT OPTION;` 3. `GRANT ALTER, DROP ON PSE MYPSE1 TO USER1, ROLE_1 WITH GRANT OPTION;`
Revoke the object, schema, or PSE privilege entirely.	1. `REVOKE <object-privilege>[,...] ON { <object-name> FROM <grantee>[,...]` 2. `REVOKE <schema-privilege>[,...] ON SCHEMA <schema-name> FROM <grantee>[,...]`

Grant Type	Statement
	3. `REVOKE <pse-privilege>[,...] ON PSE <pse-name> FROM <grantee>[,...]` Example: 1. `REVOKE SELECT, UPDATE ON TABLE1 FROM USER1, ROLE_1;` 2. `REVOKE ALTER ON SCHEMA MYSCHEMA1 FROM USER1, ROLE_1;` 3. `REVOKE ALTER, DROP ON PSE MYPSE1 FROM USER1, ROLE_1;`

Related Information

GRANT Object-Level Privilege Statement
REVOKE Object-Level Privilege Statement

4.1.3.3 Granting and Revoking the BACKUP TABLE Privilege

Grant and revoke the privilege to back up specific tables in SAP IQ.

Prerequisites

Requires one of:

- You have administrative rights on the BACKUP TABLE object-level privilege on the table.
- If you own the table, you have the BACKUP OWNER TABLE or BACKUP ANY TABLE system privilege.
- If the table is owned by others, you have the BACKUP ANY TABLE system privilege.

Procedure

To grant or revoke the BACKP TABLE privilege, execute one of the following statements:

Grant Type	Statement

Grant system privilege with no administrative rights	`GRANT BACKUP TABLE ON <table_name> TO <userID> [,...]`
	Example:
	`GRANT BACKUP TABLE ON table1 TO user1, user2;`

Grant the object privilege with full administrative rights	`GRANT BACKUP TABLE ON <table_name> TO <userID> [,...] [ WITH GRANT OPTION ]`
	Example:
	`GRANT BACKUP TABLE ON table1 TO user1, user2 WITH GRANT OPTION;`

Revoke object privilege entirely	`REVOKE BACKUP TABLE ON <table_name> FROM <userID> [,...]` `REVOKE BACKUP TABLE ON table1 FROM user1, user2;`

Related Information

GRANT BACKUP TABLE Privilege Statement
REVOKE BACKUP TABLE Privilege Statement

4.1.3.4 Granting and Revoking the RESTORE TABLE Privilege

Grant and revoke the privilege to restore specific tables in SAP IQ.

Prerequisites

Requires one of:

- You have the RESTORE TABLE object-level privilege on the table granted with the WITH GRANT option.
- If you own the table, you have the RESTORE OWNER TABLE or RESTORE ANY TABLE system privilege.
- If the table is owned by others, you have the RESTORE ANY TABLE system privilege.

Procedure

To grant or revoke the RESTORE TABLE privilege, execute one of the following statements:

Grant Type	Statement

Grant object privilege with no administrative rights.	`GRANT RESTORE TABLE ON <table_name> TO <userID> [,...]`
	Example:
	`GRANT RESTORE TABLE ON table1 TO user1, user2;`

Grant object privilege with full administrative rights.	`GRANT RESTORE TABLE ON <table_name> TO <userID> [,...] WITH GRANT OPTION`
	Example:
	`GRANT RESTORE TABLE ON table1 TO user1, user2 WITH GRANT OPTION;`

Revoke object privilege entirely	`REVOKE RESTORE TABLE ON <table_name> FROM <userID> [,...]`
	Example:
	`REVOKE RESTORE TABLE ON table1 FROM user1, user2;`

Related Information

GRANT RESTORE TABLE Privilege Statement
REVOKE RESTORE TABLE Privilege Statement

4.1.3.5 Granting and Revoking the CREATE Privilege

Grant or revoke the privilege to create database objects in the specified dbspace.

Prerequisites

Requires the MANAGE ANY DBSPACE system privilege.

Procedure

To grant the CREATE privilege, enter:

Grant Type	Statement

Grant the object or schema privilege with no administrative rights.	`GRANT CREATE ON <dbspace_name> TO <userID> [,...]`
	Example:
	`GRANT CREATE ON dbspace1 TO user1, user2;`

Grant the object privilege with full administrative rights.	`GRANT CREATE ON <dbspace_name> TO <userID> [,...] [ WITH GRANT OPTION ]`
	Example:
	`GRANT CREATE ON dbspace1 TO user1, user2 WITH GRANT OPTION;`

Revoke object privilege entirely.	`REVOKE CREATE ON dbspace1 FROM user1, user2;` Example:

Related Information

GRANT CREATE Privilege Statement
REVOKE CREATE Privilege Statement

4.1.3.6 Listing Object Privileges Granted to a User or Role

Display the object privileges granted on objects owned by a schema.

To view object privileges granted to a user, see Listing Object Privileges Granted to a User [page 30].

To view roles and system privileges granted to a role, see Listing Object Privileges Granted to a Role [page 128].

4.1.3.7 Privileges Required to Manage Table Objects in a Dbspace

The privileges required depend on the task you are performing.

To create a new table on a dbspace requires the CREATE object-level privilege on the dbspace. To move an existing table or column to a dbspace requires the MANAGE ANY DBSPACE system privilege or the CREATE object-level privilege on the destination dbspace.

In addition to the dbspace requirements, you also require a system privilege for the specific task. For example, you need the CREATE TABLE or CREATE ANY TABLE system privilege to create a table, the ALTER ANY TABLE system privilege to alter the table, and so on.

For example, to create `table1`, owned by you, in dbspace `test1`, you require the CREATE object-level privilege on `test1`, as well as the CREATE TABLE system privilege. To then move `table1` from dbspace `test1` to dbspace `test2` requires either the MANAGE ANY DBSPACE system privilege or the CREATE object-level privilege on `test2`, the destination dbspace.

You can grant the required privileges to, or revoked them from, a user or a role. Any member in a role inherits the privileges from the role.

By default, the CREATE object-level privilege on IQ_SYSTEM_MAIN, IQ_SYSTEM_TEMP, and SYSTEM is granted to PUBLIC.

4.1.3.8 Command Line Options That Control Privileges

The database server start-up command `start_iq` includes options that set the privilege level of some database and server functions.

Switches That Start and Stop Databases

The `-gd` option lets you limit the users who can start or stop a database on a running server to those with a certain level of privilege in the database to which they are already connected:

- `DBA` – (default value) only users with SERVER OPERATOR system privilege can start an extra database.
- `ALL` – (default in start_iq and `default.cfg`) any user can start and stop databases. This setting means that the DBA does not need to issue `START DATABASE` commands. Users must still be granted the privileges to access a particular database once he or she has started it.
- `NONE` – no one can start or stop a database from Interactive SQL on a running server.

> **Note**
>
> If `-gd ALL` is not set when you start the server, only a user with the SERVER OPERATOR system privilege can start additional databases on that server. This means that users cannot connect to databases that are not already started, either at the same time as the server, or since then by a user with the SERVER OPERATOR system privilege. However, it also lets a user without the SERVER OPERATOR system privilege stop a database. For this reason, you may want to change this setting to DBA on production databases.

Switches That Create and Delete Databases

The `-gu` option limits the users who can create and drop databases to those with a certain level of privilege in the database to which they are connected.

- `DBA` – only users with SERVER OPERATOR system privilege can create and drop databases.
- `ALL` (default) – any user can create and drop databases.
- `NONE` – no user can create or drop a database.
- `UTILITY_DB` – only those users who can connect to the `utility_db` database can create and drop databases.

Stop Server Switch

The `-gk` option limits the users who can shut down a server with the `dbstop` utility or `STOP ENGINE` command:

- `DBA` (default) – only users with SERVER OPERATOR system privilege can stop the server.
- `ALL` – any user can stop the server.
- `NONE` – no user can shut down the server with the `dbstop` utility or `STOP ENGINE` command.

Switches That Load and Unload Databases

The `-gl` option limits the users who can load data using `LOAD TABLE` to users with a certain level of privilege in the database.

- `DBA` – any user with the LOAD ANY TABLE, ALTER ANY TABLE, or ALTER ANY OBJECT system privilege can load data.
- `ALL` (default for `start_iq` and `default.cfg`) – any user can load data.
- `NONE` – data cannot be loaded.

4.1.4 System Procedure Privileges

There are two security models under which privileged system procedures can run. Each model grants the ability to run the system procedure differently.

> **Note**
>
> The following information applies only to SAP IQ privileged system procedures, not user-defined stored procedures.

The first model, called the SYSTEM PROCEDURE DEFINER model, runs a privileged system procedure with the privileges of its owner, typically dbo. The second model, called the SYSTEM PROCEDURE INVOKER model, runs a privileged system procedure with the privileges of the person executing it.

To run a privileged system procedure using the SYSTEM PROCEDURE DEFINER model, grant explicit EXECUTE object-level privilege on the procedure. Any system privileges required to run any underlying authorized tasks of the system procedure are automatically inherited from the owner (definer of the system procedure).

For privileged system procedures using the SYSTEM PROCEDURE INVOKER model, the EXECUTE object-level privilege is granted to the PUBLIC role, and since, by default, every user is a member of the PUBLIC role, every user automatically inherits the EXECUTE object-level privilege. However, since the PUBLIC role is not the owner of the system procedures, and is not granted any system privileges, the system privileges required to run any underlying authorized tasks must be granted directly or indirectly to the user.

By default, a database created in versions 16.x and later runs all privileged system procedures using the SYSTEM PROCEDURE INVOKER model. A database created in versions earlier than 16.x and upgraded to versions 16.x and later runs privileged system procedures using a combination of both the SYSTEM PROCEDURE DEFINER and SYSTEM PROCEDURE INVOKER models. In the combined model, all pre- 16.x privileged system procedures use the SYSTEM PROCEDURE DEFINER model, and any privileged system procedures introduced with 16.x (or any future release) use the SYSTEM PROCEDURE INVOKER model. You can override the default security model when creating or upgrading a database, or any time thereafter. However, SAP recommends that you not do so, as it may result in loss of functionality on custom stored procedures and applications.

In this section:

4.1.4.1 Granting and Revoking the Privilege to Run a Function or Procedure

Grant or revoke the privilege to execute or call a procedure or user-defined function.

Prerequisites

Requires one of:

- The MANAGE ANY OBJECT PRIVILEGE system privilege or,
- You own the procedure.

Context

The EXECUTE object-level privilege does not support the WITH GRANT OPTION clause.

Procedure

To grant or revoke the EXECUTE privilege, execute one of the following statements:

```
GRANT EXECUTE ON { <procedure_name> | <function_name> } TO <userID> [,...]
```

```
REVOKE EXECUTE ON { <procedure_name> | <function_name> } FROM <userID> [,...]
```

Example

This example grants the EXECUTE privilege on procedure `sp_test` to `user1` and `user2`:

```
GRANT EXECUTE ON sp_test TO user1, user2;
```

This example revokes the EXECUTE privilege on procedure `sp_test` from `user1` and `user2`:

```
REVOKE EXECUTE ON sp_test FROM user1, user2;
```

Related Information

GRANT Object-Level Privilege Statement
REVOKE Object-Level Privilege Statement

4.1.4.2 Determining the Security Model Used by a Database

There are two security models a database can use.

To determine the security model a database is using, execute:

```
select IF
((HEXTOINT(substring(db_property('Capabilities'),1,length(db_property('Capabiliti
es'))-20)) & 8) = 8)
    THEN 1
    ELSE 0
    END IF
```

Where:

- 1 – indicates the database is using the SYSTEM PROCEDURE INVOKER model.
- 0 – indicates that the database is using the combined model.

In the combined model, only pre-16.0 privileged system procedures run using the SYSTEM PROCEDURE DEFINER. Refer to the pre-16.0 privileged system procedures list to identify these system procedures.

You cannot configure a new or upgraded database version 16.0 or later to run all system procedures using the SYSTEM PROCEDURE DEFINER model.

4.1.4.3 Pre-16.x Privileged System Procedures

A list of pre-16.x privileged system procedures.

Privileged System Procedures Using the Combined Security Model

For these privileged system procedures, if the database is configured to use SYSTEM PROCEDURE DEFINER, you only need EXECUTE object-level privilege on the procedure to run it. If the database is configured to use SYSTEM PROCEDURE INVOKER, you also need the individual system privileges required by each procedure. Refer to *SAP IQ SQL Reference* for the system privileges required to run each system procedure.

- sa_audit_string
- sa_checkpoint_execute
- sa_disable_auditing_type
- sa_disk_free_space
- sa_enable_auditing_type
- sa_external_library_unload
- sa_flush_cache
- sa_list_external_library
- sa_server_option
- sa_procedure_profile
- sa_procedure_profile_summary
- sa_table_page_usage
- sa_validate
- sp_iq_reset_identity
- sp_iqaddlogin
- sp_iqbackupdetails
- sp_iqbackupsummary
- sp_iqcardinality_analysis
- sp_iqcheckdb
- sp_iqcheckoptions
- sp_iqclient_lookup
- sp_iqcolumn
- sp_iqcolumnuse
- sp_iqconnection
- sp_iqconstraint
- sp_iqcontext
- sp_iqconstraint
- sp_iqcontext
- sp_iqcursorinfo
- sp_iqdatatype
- sp_iqdbsize
- sp_iqdbspace
- sp_iqdbspaceinfo
- sp_iqdbspaceobjectinfo
- sp_iqdbstatistics
- sp_iqdroplogin
- sp_iqemptyfile
- sp_iqestdbspaces
- sp_iqestspace
- sp_iqevent
- sp_iqfile
- sp_iqhelp
- sp_iqindex
- sp_iqindex_alt
- sp_iqindexadvice
- sp_iqindexfragmentation
- sp_iqindexinfo
- sp_iqindexmetadata
- sp_iqindexsize
- sp_iqindexuse
- sp_iqlmconfig
- sp_iqlocks
- sp_iqmodifyadmin
- sp_iqmodifylogin
- sp_iqmpxcheckdqpconfig
- sp_iqmpxdumptlvlog
- sp_iqmpxfilestatus
- sp_iqmpxincconnpoolinfo
- sp_iqmpxincheartbeatinfo
- sp_iqcopyloginpolicy
- sp_iqmpxincconnpoolinfo
- sp_iqmpxincheartbeatinfo
- sp_iqmpxinfo
- sp_iqmpxversioninfo
- sp_iqobjectinfo
- sp_iqpkeys
- sp_iqprocedure
- sp_iqprocparm
- sp_iqrebuildindex
- sp_iqrename
- sp_iqrestoreaction
- sp_iqrowdensity
- sp_iqsetcompression
- sp_iqsharedtempdistrib
- sp_iqshowcompression
- sp_iqshowpsexe
- sp_iqspaceinfo
- sp_iqspaceused
- sp_iqstatistics
- sp_iqstatus
- sp_iqsysmon
- sp_iqtable
- sp_iqtablesize
- sp_iqtableuse
- sp_iqtransaction
- sp_iqunusedcolumn
- sp_iqunusedindex
- sp_iqunusedtable
- sp_iqversionuse
- sp_iqview
- sp_iqwho
- sp_iqworkmon

Privileged System Procedures Using Invoker Privileges

These pre-16.x privileged system procedures run with the privileges of the user who is running the procedure, not the owner of the procedure, regardless of the security model setting. Therefore, in addition to the EXECUTE object-level privilege on the system procedure, which is, by default, granted through membership in PUBLIC role, you must also be granted the additional system privileges required by the system procedure. See the *SAP IQ SQL Reference* for the system privileges required to run each system procedure.

- sa_describe_shapefile
- sa_get_user_status
- sa_locks
- sa_performance_diagnostics

- sa_report_deadlocks
- sa_text_index_stats

4.2 Roles

A role is a container that contains system privileges, object privileges, and other roles. Granting privileges to and from a role and then granting the role to a user is semantically equivalent to granting the privileges directly to a user.

A role name must be unique and cannot match a user name. Unless stated otherwise, the term user-defined role includes both user-defined roles and user-extended roles.

The types of roles are as follows:

- User-Defined Roles [page 91]
- User-extended Roles [page 92]
- System Roles [page 96]

In this section:

Role-Based Security [page 88]
: Role-based security is an access control model where permissions are assigned to roles rather than directly to individual users.

User-Defined Roles [page 91]
: A user-defined role is a custom collection of system and object-level privileges and roles, typically created to group privileges that are related to a specific task or set of tasks.

User-extended Roles [page 92]
: A user-extended role is a user ID that acts as a role and is grantable to others.

System Roles [page 96]
: System roles are built-in roles that each new database creates automatically.

Compatibility Roles [page 104]
: Compatibility roles exist for backward compatibility with versions of SAP IQ earlier than 16.0 that supported authority-based security.

Role and Global Role Administrators [page 104]
: Roles are managed by role administrators or global role administrators. Administrators are responsible for granting and revoking user-defined roles. Global role administrators are the sole administrators of system roles.

Managing Roles [page 116]
: Add and remove users, and system, object, and schema privileges to and from roles.

Referencing Objects Owned by User_Extended Roles [page 131]
: Views, procedures, and tables are more easily managed when they are owned by a user-extended role instead of a user.

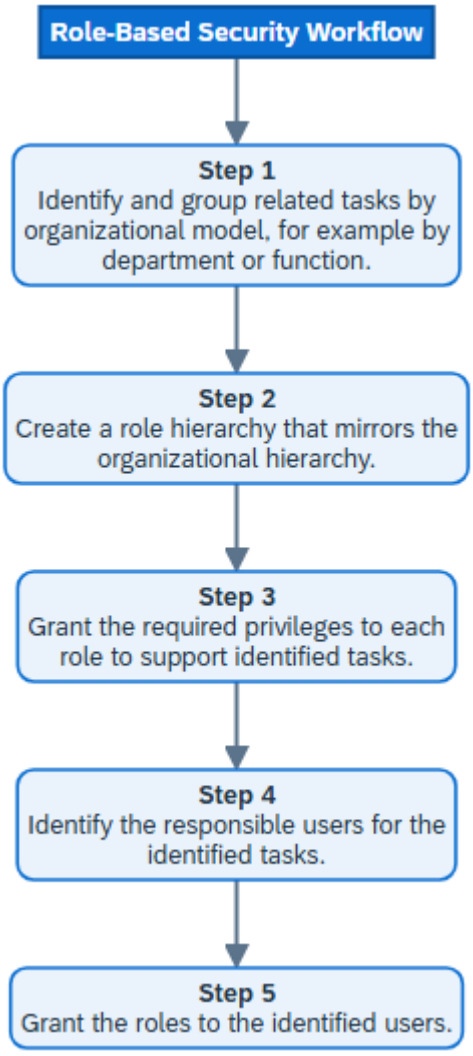
Determining the Roles and Privileges Granted to a User [page 132]
: The `sp_has_role` stored function returns an integer value that indicates whether the invoker of the procedure has been granted the specified system privilege or user-defined role.

4.2.1 Role-Based Security

Role-based security is an access control model where permissions are assigned to roles rather than directly to individual users.

Each role represents a set of related tasks or responsibilities — such as a department or job function — and users are granted access by being assigned the appropriate role. This simplifies administration, reduces the risk of over-privileged accounts, and makes auditing easier since access is managed at the role level rather than per user.

The workflow for designing roles-based security starts by organizing related work into clear task groups, such as by department or function, so responsibilities are logically clustered. From there, a role hierarchy is built to mirror the organization's structure, making delegation and governance easier to manage. Each role is then assigned the specific privileges and any subordinate roles it needs to perform those tasks effectively, and finally the roles are granted to users so they can carry out their responsibilities with appropriate access.



For more information on designed role-based security, see TUTORIAL: Designing Role-Based Security [page 89].

In this section:

4.2.1.1 TUTORIAL: Designing Role-Based Security

This tutorial is designed to walk you through the analysis to begin developing a role-based security plan.

Context

This analysis is meant as a guideline only, a starting point upon which to build your plan. It is assumed that you understand the concept of role-based security.

Procedure

1. Identify tasks to be carried out. Review the User Management [page 9], Privileges [page 48], Roles [page 87], and Authentication [page 136] sections in this guide to identify relevant tasks to designing role-based security. Suppose you identify the following tasks:
   - Create new users.
   - Change user passwords.
   - Create and manage login policies.
   - Assign login policies to users.
   - Create and manage roles.
   - Assign roles to users.
   - Grant and revoke privileges to users and roles.
   - Create and manage procedures.
   - Create and manage certificates and credentials.
   - Set database options.
2. Group the list into related tasks. Give each group a name, which will become the name of a role. The name should be something that identifies the kinds of tasks assigned to the role.

	user_mgt	access_ctrl	session_mgt	db_ctrl
Create new users.	x			
Change user passwords.	x			
Create and manage login policies.		x		
Assign login policies to users.		x		

	user_mgt	access_ctrl	session_mgt	db_ctrl
Create and manage roles.	x			
Assign roles to users.	x			
Grant and revoke privileges to users and roles.	x			
Create and manage certificates and credentials.			x	
Set database options.				x

3. For each group, use the SAP IQ Administration: User Management and Security and SAP IQ SQL Reference guides to identify the SQL statements required to perform each task and list the documented privileges required for each statement.

Group	Task	SQL Statement	Required Privilege
user_mgt	Create and delete users.	• `CREATE USER` • `DROP USER`	• `USER MANAGEMENT`
	Change user passwords.	• `ALTER USER`	• `CHANGE PASSWORD`
	Assign and de-assign login policies to users.	• `ALTER USER`	• `MANAGE ANY LOGIN POLICY`
	Create and manage roles.	• `CREATE ROLE` • `DROP ROLE`	• `MANAGE ROLES`
	Assign roles to users.	• `GRANT ROLE` • `REVOKE ROLE`	• `MANAGE ROLES`
	Assign privileges to users and roles.	• `GRANT <system privilege>`	• Administrative rights on the privilege being granted.
access_ctrl	Create and manage login policies.	• `CREATE LOGIN POLICY` • `ALTER LOGIN POLICY` • `DROP LOGIN POLICY`	• `MANAGE ANY LOGIN POLICY`
	Assign login policies to users.	• `ALTER USER`	• `MANAGE ANY LOGIN POLICY`
session-mgt	Create and manage certificates.	• `CREATE CERTIFICATE` • `DROP CERTIFICATE` • `CREATE CREDENTIAL` • `DROP CREDENTIAL` • `CREATE PSE` • `DROP PSE`	• `MANAGE OWNER CERTIFICATES` • `MANAGE CERTIFICATES`

Group	Task	SQL Statement	Required Privilege
db_ctrl	Set database options	• `SET OPTION`	• `SET ANY CUSTOMER PUBLIC OPTION` • `SET ANY CUSTOMER SYSTEM OPTION` • `SET ANY CUSTOMER SECURITY OPTION`

4. Decide who will perform the various tasks. Consider a tiered approach to the groupings. For example, USER1 performs tasks in group 1. USER2 performs tasks in group 2. USER1 and USER2 are managed by USER3. As their manager, USER3 should be able to perform all the tasks of both users.

	USER1	USER2	USER3	USER4	USER5
user_mgt	x		x		x
access_ctrl		X	x		x
session_mgt				x	x
dDb_ctrl					x

Next Steps

You've now identified the roles required, the privileges to be granted to each, and which users will be granted the roles to implement your role-based security system.

For this tutorial, once implemented,

- USER1 can perform tasks for user management.
- USER2 can manage login policies.
- USER3 can manage both users and login policies.
- USER4 can manage certificates and credentials.
- USER5 can manage all the above.

4.2.2 User-Defined Roles

A user-defined role is a custom collection of system and object-level privileges and roles, typically created to group privileges that are related to a specific task or set of tasks.

The granting of a user-defined role to a user or another role is semantically equivalent to granting each underlying system privilege, object privilege, and subordinate roles individually.

User-defined roles can own objects.

A role can be granted to a user or another role with:

1. Administrative rights
2. Administrative rights only
3. No administrative rights

With administrative rights, the recipient can manage the role by granting and revoking membership and by dropping the role. The recipient can also use the role's underlying privileges and subordinate roles.

With administrative rights only, the recipient can manage the role, but cannot use the role's underlying privileges and subordinate roles.

With no administrative rights, the recipient can use the underlying privileges and roles of the role, but cannot grant or revoke membership or drop the role.

A special type of user-defined role is the user-extended role: a user that has been extended to function as a role and can be granted to others. A user-defined role cannot be converted to a user-extended role.

For each role, the system must always maintain at least the minimum number of role or global role administrators with login passwords, as defined by the `MIN_ROLE_ADMINS` database option. If removing a role member who is an administrator would violate this minimum, the removal fails. See Role and Global Role Administrators [page 104].

4.2.3 User-extended Roles

A user-extended role is a user ID that acts as a role and is grantable to others.

The user-extended role retains the login privileges of the original user

The user who became a user-extended role can administer the new role (grant and revoke it to others) unless this privilege is explicitly removed.

You can grant a user-extended role to another user or role. The grantee inherits all the system and object privileges of the user-extended role, including any associated administration rights.

Because ownership of database objects is associated with a single user ID, when the owner is a user-extended role, ownership of the database object is not inherited by its grantees. Only granted privileges are inherited.

User-extended roles are managed by role and global role administrators. A minimum number of role or global role administrators (as defined by the MIN_ROLE_ADMINS database option) with a login password must exist for each role at all times. If removing a role member who is an administrator would violate the minimum requirement, the removal fails. See Role and Global Role Administrators [page 104].

In this section:

4.2.3.1 Converting a User to a User-Extended Role

Extend an existing user ID to act as a role. User-extended roles let you grant the set of system and object privileges and roles of a user to other users and roles.

Prerequisites

- Requires the `MANAGE ROLES` system privilege

Context

Extending a user to act as a role is semantically equivalent to creating a user-defined role, granting the role a set of privileges and roles, and then granting the role to other users.

User-extended roles retain the login privileges of the originating user. Like regular roles, user-extended roles have role or global role administrators. They are designated during conversion or later. If no role administrators are designated, the role is managed by global role administrators.

A minimum number of role or global role administrators (as defined by the `MIN_ROLE_ADMINS` database option) must be maintained for all user-extended roles.

Procedure

To convert an existing user, execute one of the following statements:

Administrative Type of User-extended Role	Syntax

Allow only global role administrators to manage the role.	`CREATE ROLE FOR USER <userID>`
	Example:
	`CREATE ROLE FOR USER USER1;`

Allow specific role administrators to administer the role but do not make them members of the role.	`CREATE ROLE FOR USER <userID> WITH ADMIN ONLY`
	Example:
	`CREATE ROLE FOR USER USER2 WITH ADMIN ONLY USER1, USER2;`

Allow specific role administrators to administer the role and be members of the role.	`CREATE ROLE FOR USER <userID> WITH ADMIN`

Administrative Type of User-extended Role	Syntax
Note **The SYS_MANAGE_ROLES_ROLE role cannot be designated a role administrator when the WITH ADMIN clause is used.**	Example: `CREATE ROLE FOR USER USER3 WITH ADMIN USER1, USER2;`

Allow specific role administrators and global role administrators to manage the role, but not become members of the role.	`CREATE ROLE FOR USER <userID> WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE`
	Example:
	`CREATE ROLE FOR USER USER4 WITH ADMIN ONLY USER1, USER2, SYS_MANAGE_ROLES_ROLE;`

Related Information

Role and Global Role Administrators [page 104]
CREATE ROLE Statement
Minimum Number of Role Administrators [page 105]

4.2.3.2 Converting a User-Extended Role Back to a User

You can convert a user-extended role back to a regular user.

Prerequisites

- Requires the `MANAGE ROLES` system privilege

Context

Once converted, the user retains any privileges and roles granted to the user-extended role. The user remains the owner of any objects created after the user was extended. Any members of the user-extended role are revoked during conversion.

Procedure

To convert a user-extended role back to a user, execute one of the following:

Convert Condition	Statement
Extended role has not been granted any members.	`DROP ROLE FROM USER <role_name>`
Extended role has been granted members.	`DROP ROLE FROM USER <role_name> WITH REVOKE`

Related Information

DROP ROLE Statement

4.2.3.3 Listing Members of a User-Extended Role

Display all users and roles granted membership in a role.

Prerequisites

No additional privileges are required.

Procedure

To display all members, execute:

```
SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
   WHERE role_name ='<user-extended-role-name>'
```

Where:

grant_type	How role granted
5	Granted with `WITH NO ADMIN OPTION` clause (role only)
6	Granted with `WITH ADMIN ONLY OPTION` clause
7	Granted with `WITH ADMIN OPTION` clause

Example

This example lists all members of role ROLE1:

```
SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
   WHERE role_name ='USER1';
```

role_name	grantee_name	grant_type	grantor
USER1	ROLE2	5	USER1
USER1	SYS_MANAGE_ROLES_ROLE	6	USER1
USER1	USER2	7	USER1
USER1	USER3	5	USER1

ROLE2 and USER3 are granted role membership only (5). SYS_MANAGE_ROLES_ROLE and USER1 are granted role administrative rights only (6). USER2 is granted role membership and administrative rights (7).

4.2.4 System Roles

System roles are built-in roles that each new database creates automatically.

A system role can't be dropped. Its underlying system privileges can't be changed. You can't add or remove additional privileges to or from a system role. System roles can't be granted with administrative rights.

System roles have no password assigned. Therefore, users can't connect to an instance as a system role. System roles don't own objects.

Revoking a user's membership in an automatically granted system role can negatively impact their functionality, and is strongly discouraged.

There are three types of system roles:

PUBLIC System Role

The PUBLIC role is automatically granted to every new user, with no administrative rights. Membership in this role allows you to see information about the database schema and allows you to execute many, but not all, system procedures and functions.

SYS System Role

The SYS role is automatically granted to every new user, with no administrative rights. Membership in this role allows you access to data from some system views.

SYS_AUTH_DBA_ROLE The SYS_AUTH_DBA_ROLE role contains all available system and object privileges. When granted to a user, the user becomes equivalent in privilege to the DBA user.

In this section:

Granting the dbo System Role [page 97]
The dbo system role owns many system stored procedures and views.

Granting the diagnostics System Role [page 98]
Members of the diagnostics system role inherit SELECT, INSERT, UPDATE, DELETE, and ALTER privileges on diagnostic tables and views.

4.2.4.1 Granting the dbo System Role

The dbo system role owns many system stored procedures and views.

Prerequisites

MANAGE ROLES system privilege.

Context

By default, the dbo system role is a member of the SYS system role and the SYS_AUTH_RESOURCE_ROLE compatibility role with no administrative rights. It is also a member of the SYS_AUTH_DBA_ROLE compatibility role with full administrative rights.

You can grant the dbo system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the dbo system role.

You can grant and revoke system privileges and roles to and from the dbo system role..

Procedure

To grant the dbo system role, execute:

```
GRANT ROLE dbo TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.2 Granting the diagnostics System Role

Members of the diagnostics system role inherit SELECT, INSERT, UPDATE, DELETE, and ALTER privileges on diagnostic tables and views.

Prerequisites

MANAGE ROLES system privilege.

Context

You can grant the diagnostics system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the diagnostics system role.

You can grant and revoke system privileges and roles to and from the diagnostics system role.

Procedure

To grant the diagnostics system role, execute:

```
GRANT ROLE diagnostics TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.3 Granting the PUBLIC System Role

The PUBLIC system role has the SELECT object privilege granted on a set of system tables and EXECUTE object privilege on system procedures.

Prerequisites

Requires:

- MANAGE ROLES system privilege

Context

The PUBLIC system role is a member of the dbo and SYS system roles, without administrative rights. As a member of the SYS role, it has read access for some system tables and views, which allows any user of the database to see information about the database schema. To restrict this access, revoke PUBLIC's membership in the SYS system role.

Any new user ID is automatically a member of the PUBLIC system role and inherits any privileges that are specifically granted to that role. Although you can remove a user from the PUBLIC system role, SAP recommends that you do not, as doing so might impact a user's ability to run system stored procedures.

You can grant the PUBLIC system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the PUBLIC system role.

You can grant and revoke system privileges and roles to and from the PUBLIC system role.

Procedure

To grant the PUBLIC system role, execute:

```
GRANT ROLE PUBLIC TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.4 Granting the rs_systabgroup System Role

The rs_systabgroup system role owns tables and system procedures that are required for Replication Server, and contains the required 0system privileges to perform Replication Server functionality.

Prerequisites

Requires:

- MANAGE ROLES system privilege

Context

You can grant the rs_systabgroup system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the rs_systabgroup system role.

You can grant and revoke system privileges and roles to and from the rs_systabgroup system role.

Procedure

To grant the rs_systabgroup system role, execute:

```
GRANT ROLE rs_systabgroup TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.5 Granting the SYS System Role

The SYS system role owns the system tables and views for the database. These tables contain the full description of the database schema, including all database objects and user IDs.

Prerequisites

Requires:

- MANAGE ROLES system privilege

Context

The SYS system role is granted the dbo and PUBLIC system roles without administrative rights. However, members of the dbo and PUBLIC system roles do not inherit any system privileges directly or indirectly from the SYS system role.

You can grant the SYS system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the SYS system role.

You cannot grant or revoke additional system privileges to or from the SYS system role.

Procedure

To grant the SYS system role, execute:

```
GRANT ROLE SYS TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.6 Granting the SYS_OFFLINE_RESET_PASSWORD_ROLE System Role

The SYS_OFFLINE_RESET_PASSWORD_ROLE lets a super-user reset the DBA password in the event of a lost password.

Prerequisites

- A super-user exists and has system access that allows SAP IQ server startup and shutdown.

Context

The SYS_OFFLINE_RESET_PASSWORD_ROLE cannot be granted with administrative rights.

Procedure

As DBA, grant SYS_OFFLINE_RESET_PASSWORD_ROLE to a super-user:

```
GRANT ROLE SYS_OFFLINE_RESET_PASSWORD_ROLE TO <grantee [,...]>
```

4.2.4.7 Granting the SYS_SPATIAL_ADMIN_ROLE System Role

The SYS_SPATIAL_ADMIN_ROLE system role grants users the ability to create, alter, drop, or comment on spatial reference systems and spatial units of measure. The SYS_SPATIAL_ADMIN_ROLE system role is the owner of all spatial objects.

Prerequisites

Requires:

- MANAGE ROLES system privilege

Context

The SYS_SPATIAL_ADMIN_ROLE system role is granted the MANAGE ANY SPATIAL OBJECT system privilege with no administrative rights.

You can grant the system role to other roles only without administrative rights, using the WITH NO ADMIN OPTION clause. The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for the SYS_SPATIAL_ADMIN_ROLE system role.

You can grant and revoke system privileges and roles to and from the SYS_SPATIAL_ADMIN_ROLE system role.

Procedure

To grant the SYS_SPATIAL_ADMIN_ROLE system role, execute:

```
GRANT ROLE SYS_SPATIAL_ADMIN_ROLE TO <grantee [,...]>
```

Related Information

GRANT ROLE Statement

4.2.4.8 Revoking a System Role

Revokes a system role from a user or role.

Prerequisites

Requires:

- Administrative privilege over the system role being revoked.

Procedure

To revoke a system role, execute:

```
REVOKE ROLE <role_name> FROM <grantee [,...]>
```

Example

This statement revokes the dbo system role entirely from `USER1`:

```
REVOKE ROLE dbo FROM USR1;
```

Related Information

REVOKE ROLE Statement

4.2.5 Compatibility Roles

Compatibility roles exist for backward compatibility with versions of SAP IQ earlier than 16.0 that supported authority-based security.

You can grant, revoke, and under specific conditions, deleted compatibility roles. You cannot modify any of the underlying system privileges of a compatibility role. However, you can migrate compatibility roles to user-defined roles, and then modify the underlying system privileges. When you migrate a compatibility role, all grantees of the compatibility role are automatically granted the user-defined role.

See *Conceptual Changes Between 15.4.x and 16.x > Role-Based Security Model > Compatibility Roles* in the SAP IQ Installation and Update Guide appropriate to your operating system.

4.2.6 Role and Global Role Administrators

Roles are managed by role administrators or global role administrators. Administrators are responsible for granting and revoking user-defined roles. Global role administrators are the sole administrators of system roles.

There is no limit to the number of role administrators that can be granted to a single role. However, there is a minimum number, as specified by the `MIN_ROLE_ADMINS` database option. This minimum requirement is validated before you can revoke a role administrator or a global role administrator from a role. The minimum number of role administrators can be set to any value between 1 (default) and 10. If you plan to increase the minimum value, you should do so before creating roles because changing this value after roles exist may be restricted. The number of role administrators defined for all roles must meet the new value before you can change it. See Minimum Number of Role Administrators [page 105].

A role administrator can be a user, a user-extended role, or a user-defined role. When designating role administrators, the `ADMIN` clause controls whether the role administrators are also members of the role, inheriting all roles and privileges of the role (`WITH ADMIN`) or whether they can only manage the role (`WITH ADMIN ONLY`).

Role administrators can be designated when you create a role or can be added later (default). If no role administrator is designated when creating a role, then global role administrators administer the role. Global role administrators are any users granted the `MANAGE ROLES` system privilege.

If role administrators are specified at role creation, then global role administrators cannot administer the role unless the `SYS_MANAGE_ROLES_ROLE` is explicitly granted to the role. For this reason, SAP recommends that you always include `SYS_MANAGE_ROLES_ROLE` in the list of role administrators when designating role administrators during role creation.

In this section:

4.2.6.1 Minimum Number of Role Administrators

The `MIN_ROLE_ADMINS` database option is a configurable value that ensures you never create a scenario where there are no users and roles left with sufficient system privilege to manage the remaining users and roles.

This value applies to the minimum number of role administrators for each role, not for the total number of roles, and is considered when you:

- Create or revoke roles.
- Drop users or roles.
- Change a user's password to null. Users or roles without passwords cannot be administrators.

To be a role administrator, users and roles must have a password.

When you attempt to change the `MIN_ROLE_ADMINS` value, the system validates that each existing role continues to have at least as many role administrators as defined by the new value. If even one role fails to meet this requirement, the statement fails. Similarly, when dropping users, if the number of remaining administrators drops below the designated minimum value, the statement fails. Locked accounts are not considered when counting the number of administrators for a role.

Before you can increase the minimum value, you need to add additional role administrators to any roles with fewer administrators than the new value. Only then can you successfully modify the `MIN_ROLE_ADMINS` value.

Example 1

- `MIN_ROLE_ADMINS` value is 2.
- `ROLE1` has two administrators and `ROLE2` has three administrators.

You can reduce the `MIN_ROLE_ADMINS` value to 1 because both roles would still meet the new minimum value. But you cannot increase the value to 3 because `Role1` would no longer have sufficient administrators to meet the new minimum value.

Example 2

- `MIN_ROLE_ADMINS` value is 2.
- `ROLE1` has two administrators, `USER1` and `USER2`.

You cannot drop `USER1` because doing so would in `ROLE1` having insufficient role administrators to meet the minimum value. You would need to designate another role administrator for `ROLE1` before you could drop `USER1`

In this section:

Setting the Minimum Number of Role Administrators [page 106]
Set the minimum number of role administrators required to manage each role.

Related Information

Automatic Unlocking of User Accounts [page 143]
MIN_ROLE_ADMINS Option

4.2.6.1.1 Setting the Minimum Number of Role Administrators

Set the minimum number of role administrators required to manage each role.

Prerequisites

Requires:

- `SET ANY CUSTOMER SECURITY OPTION` system privilege

Context

The minimum number of role administrators is an integer between 1 (default) and 10. You cannot change this value if doing so results in the number of role administrators for any single role not meeting the new minimum value. You also cannot temporarily set this option.

Procedure

To change the minimum number of role administrators, execute:

```
SET OPTION Public.min_role_admins = <value>
```

Related Information

MIN_ROLE_ADMINS Option

4.2.6.2 Troubleshooting: DBA-Equivalent User Is Unable to Administer a Role

Troubleshoot why the DBA or DBA-equivalent user cannot manage a role.

Prerequisites

Requires:

- Administrative rights over the role

Context

This behavior may occur because:

- The global role administrator role (`SYS_MANAGE_ROLES_ROLE`) has not been explicitly granted to the role and role administrators were designated during role creation.
- The DBA-equivalent user is not designated as a role administrator of the role and `SYS_MANAGE_ROLES_ROLE` has not been explicitly granted to the role.

Procedure

1. Explicitly designate the global role administrator role (SYS_MANAGE_ROLES_ROLE) as a role administrator of the role.

   ```
   GRANT ROLE <role_name> TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION;
   ```

2. Make the DBA-equivalent user a role administrator on the role.

   ```
   GRANT ROLE <role_name> TO <DBA_equivalent> WITH ADMIN ONLY OPTION;
   ```

4.2.6.3 Adding or Removing Role Administrators from a Role

Add or remove role administrators as long as the minimum number of administrators on the role meets the MIN_ROLE_ADMIN value.

Prerequisites

Requires one of the following:

- If the role has no role administrators assigned, then you are a global role administrator.
- If the role has at least one role administrator assigned, then you have administrative rights on the role.

Context

If a role was initially created with specified role administrators, then global role administrators cannot administrator the role unless the global role administrator role (SYS_MANAGE_ROLES_ROLE) is explicitly granted to the role.

When explicitly granting SYS_MANAGE_ROLES_ROLE to a role, the WITH ADMIN ONLY OPTION clause must be specified. The WITH ADMIN OPTION clause is not supported.

When removing a role administrator, if they are also a member of the role, you can remove their ability to manage the role, but retain their inheritance of the underlying system privileges and roles of the role.

Procedure

To add or remove role administrators, execute one of these statements:

Grant Type	Statement

Add a role administrator with administrative privileges only.	`GRANT ROLE <role_name> TO <role_admin_name>[,...] WITH ADMIN ONLY OPTION`
	Example:
	`GRANT ROLE ROLE_1 TO USER1, USER2 WITH ADMIN ONLY OPTION;`

Add a role administrator with administrative privilege and role membership.	`GRANT ROLE <role_name> TO <role_admin_name>[,...] WITH ADMIN OPTION`
	Example:
	`GRANT ROLE ROLE_1 TO USER1, USER2 WITH ADMIN OPTION;`

Explicitly allow global role administrators to manage a role.	`GRANT ROLE <role_name> TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION`
	Example:
	`GRANT ROLE ROLE_1 TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION;`

Remove a role administrator's administrative privilege, but retain their role membership.	`REVOKE ADMIN OPTION FOR ROLE <role_name> FROM <admin_name>[, ...]`
	Example:
	`REVOKE ADMIN OPTION FOR ROLE ROLE_1 FROM USER1;`

Remove a role administrator's administrative privilege and role membership.	`REVOKE ROLE <role_name> FROM <admin_name>[, ...]`
	Example:
	`REVOKE ROLE ROLE_1 FROM USER2;`

Explicitly remove global role administrator's from a role.	`REVOKE ROLE <role_name> FROM SYS_MANAGE_ROLES_ROLE`
	Example:
	`REVOKE ROLE ROLE_1 FROM SYS_MANAGE_ROLES_ROLE;`

Related Information

GRANT ROLE Statement
REVOKE ROLE Statement

4.2.6.4 Adding or Removing Global Role Administrator

Allow or stop a user or role from acting as a global role administrator.

Prerequisites

Requires one of:

- Administrative rights on the `MANAGE ROLES` system privilege.

Context

Global role administrators are users or roles granted the `MANAGE ROLES` system privilege. When this privilege is granted, the user or role is automatically added to the `SYS_MANAGE_ROLE_ROLE` role. No administrative privilege on `MANAGE ROLES` is required, because it is automatically inherited through `SYS_MANAGE_ROLE_ROLE`. However, administrative rights on `MANAGE ROLES` is required to grant that system privilege to another user or role.

Granting `MANAGE ROLES` to a user or role is symptomatically equivalent to granting `SYS_MANAGE_ROLE_ROLE`to a user or role .

Procedure

To allow or stop users or roles from acting as global role administrators, execute one of the following statements:

Grant Type	Statement
Allow a user or role to act as a global role administrator.	

Grant the `MANAGE ROLES` system privilege to a user or role.	`GRANT MANAGE ROLES TO <grantee>[,...]`
	Example:
	`GRANT MANAGE ROLES TO USER1, ROLE1;`

Grant Type	Statement

Add a user or role to the SYS_MANAGE_ROLES_ROLE role.	`GRANT ROLE SYS_MANAGE_ROLES_ROLE TO <grantee>[,...] WITH ADMIN ONLY OPTION`
	Example:
	`GRANT ROLE SYS_MANAGE_ROLES_ROLE TO USER1, ROLE1 WITH ADMIN ONLY OPTION;`

Stop a user or role from acting as a global role administrator.	

Revoke the MANAGE ROLES system privilege from a user or role.	`REVOKE MANAGE ROLES FROM <grantee>[,...]`
	Example:
	`REVOKE MANAGE ROLES FROM USER1, ROLE1;`

Remove a user or role from the SYS_MANAGE_ROLES_ROLE role.	`REBVOKE ROLE SYS_MANAGE_ROLES_ROLE FROM <grantee>[,...]`
	Example:
	`REVOKE ROLE SYS_MANAGE_ROLES_ROLE FROM USER1, ROLE1;`

Related Information

GRANT System Privilege Statement
REVOKE System Privilege Statement

4.2.6.5 Replacing Existing Role Administrators on a Role

Replace current role administrators with new administrators.

Prerequisites

Requires:

- Administrative rights on the role.

Context

Depending on the scope of the replacement, there are two approaches available. Each approach has different net effects on role and global administrators.

Approach 1

The first approach allows you to selectively replace the administrators of an existing role. Add new role administrators, then remove existing administrators from the role. The net of your changes must meet the minimum number of administrators requirement throughout.

Approach 2

The second approach allows you to completely replace all existing role administrators. Using the second approach also replaces the global role administrator. All current role administrators are overwritten with new role administrators. If any current role administrators are to continue in this capacity, then you must include them in the list of replacement administrators. The list replaces all existing administrators, with the following behavior:

- All existing role administrators with `WITH ADMIN OPTION` rights that are not included on the new list retain membership in the role but no longer have administrative rights on the role.
- All existing role administrators granted with `WITH ADMIN ONLY OPTION` rights that are not included on the new list are removed as administrators of the role.
- An existing role administrator included on the new list retains their original administrative rights if they are higher than the replacement rights. For example, new role administrators have `WITH ADMIN ONLY` rights. Role administrators were originally granted the role with `WITH ADMIN OPTION` rights. Original role administrators included on the new list, retain the higher `WITH ADMIN OPTION` rights. New role administrators have `WITH ADMIN ONLY OPTION` rights.
- If the global role administrator was explicitly granted to the role as a role administrator, it is removed from the role unless it is again explicitly included on the new list.
- If the `SYS_MANAGE_ROLES_ROLE` role is included on the list, and new role administrators are granted `WITH ADMIN OPTION` rights , then the `SYS_MANAGE_ROLES_ROLE` role is ignored. The `SYS_MANAGE_ROLES_ROLE` role cannot be designated as a role administrator if the `WITH ADMIN OPTION` clause is specified.

You can modify role administrators as long as the replacement administrative option is equal to or higher than the current level. To lower the administrative level, first revoke all role administrators from the role, and then regrant them.

A minimum number of role or global role administrators (as defined by the `MIN_ROLE_ADMINS` database option) with a login password must exist for each role at all times. When replacing role administrators, if the number of replacement administrators violates the minimum requirement, the replacement fails.

> → Tip
>
> To maintain the minimum number of role administrators, always add new administrators before removing existing ones.

Procedure

To replace role administrators, execute one of the following statements:

Replacement Option	Statement
Approach 1: Selective replacement	

Replace specific existing role administrators with new administrators. New administrators have administrative only rights, no role membership.	1. Add the new role administrators.
	`GRANT ROLE <role_name> TO <new_admin_name>[,...] WITH ADMIN ONLY OPTION`
	2. Then remove the old role administrators.
	`REVOKE ADMIN OPTION FOR ROLE <role_name> FROM <esisting_admin_name>[,...]`
	Example:
	1. Add USER1 and ROLE2 as new role administrators.
	`GRANT ROLE ROLE1 TO USER1, ROLE2 WITH ADMIN ONLY OPTION;`
	2. Then remove USER3 and ROLE3 as role administrators.
	`REVOKE ADMIN OPTION FOR ROLE ROLE1 FROM USER3, ROLE3;`

Replace specific existing role administrators with new administrators. New administrators have role membership and administrative rights.	1. Add the new role administrators.
	`GRANT ROLE <role_name> TO <new_admin_name>[,...] WITH ADMIN OPTION`
	2. Then remove the old role administrators.
	`REVOKE ADMIN OPTION FOR ROLE <role_name> FROM <existing_admin_name>[,...]`
	Example:
	1. Add USER1 and ROLE2 as new role administrators.
	`GRANT ROLE ROLE1 TO USER1, ROLE2 WITH ADMIN OPTION;`
	2. Then remove USER3 and ROLE3 as role administrators.
	`REVOKE ADMIN OPTION FOR ROLE ROLE1 FROM USER3, ROLE3;`

Approach 2: Replace all role administrators	

Replacement Option	Statement

Replace all existing role and global role administrators with new role administrators. New administrators have administrative rights and role membership. Global role administrators cannot manage the role.	`CREATE OR REPLACE ROLE <role_name> WITH ADMIN <admin_name>[,...]`
	Example:
	`CREATE OR REPLACE ROLE ROLE1 WITH ADMIN USER1, ROLE2;`

Replace all existing role and global role administrators with new role and global role administrators. New role and global role administrators have administrative rights, but not role membership.**	`CREATE OR REPLACE ROLE <role_name> WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, <admin_name>[,...]`
	Example:
	`CREATE OR REPLACE ROLE ROLE1 WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, USER1, ROLE2;`

Replace all existing role and global administrators with new role administrators. New administrators have administrative rights and role membership.
Then regrant the global role administrator to the role with administrative rights only so that global role administrators can continue to manage the role.**
	1. Replace the existing role administrators.
	`CREATE OR REPLACE ROLE <role_name> WITH ADMIN <admin_name>[,...]`
	2. Then regrant the global role administrator to the role with administrative rights only
	`GRANT ROLE <role_name> TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION`
	Example:
	1. Replace existing role administrators with USER1 and ROLE2.
	`CREATE OR REPLACE ROLE ROLE1 WITH ADMIN USER1, ROLE2;`
	2. Explicitly regrant the global role administrator role.
	`GRANT ROLE ROLE1 TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION;`

**`SYS_MANAGE_ROLES_ROLE` can be granted to a role only using the `WITH ADMIN ONLY` option. Therefore, when the `CREATE OR REPLACE` statement includes the `WITH ADMIN ONLY` option, `SYS_MANAGE_ROLES_ROLE` can be included in the administrator list. When the `CREATE OR REPLACE` statement uses the `WITH ADMIN` option, you must issue a separate grant statement to grant `SYS_MANAGE_ROLES_ROLE` to the role using the `WITH ADMIN ONLY` option.

Related Information

GRANT ROLE Statement
REVOKE ROLE Statement

CREATE ROLE Statement

4.2.6.6 Listing Role and Global Role Administrators

Display a list of all users and roles acting as global role administrators.

Prerequisites

No additional privileges are required.

Context

A user or role must have administrative rights on a role to act as role administrator. But a user or role doesn't need administrative rights to act as a global role administrator.

grant_type	How role granted
6	Granted with `WITH ADMIN ONLY OPTION` clause.
7	Granted with `WITH ADMIN OPTION` clause.

Procedure

1. To display all role administrators on a role, execute:

   ```
   SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
      WHERE role_name ='<role-name>' AND grant_type <> 5;
   ```

2. To display all global role administrators on a role, execute:

   ```
   SELECT role_name, grantee_name FROM SYSROLEGRANTS
      WHERE role_name ='SYS_MANAGE_ROLES_ROLE' AND grantor <> 'SYS';
   ```

   Where:

Example

This example lists all role administrators for `ROLE1`:

```
SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
   WHERE role_name ='ROLE1' AND  grant_type <> 5;
```

role_name	grantee_name	grant_type	grantor
ROLE1	SYS_MANAGE_ROLES_ROLE	6	USER1
ROLE1	USER1	6	USER1
ROLE1	USER2	7	USER1

USER1 and USER2 and all members of the SYS_MANAGE_ROLES_ROLE role are role administrators.

This example lists all global role administrators of ROLE1:

```
SELECT role_name, grantee_name FROM SYSROLEGRANTS
   WHERE role_name ='SYS_MANAGE_ROLES_ROLE' AND  grantor <> 'SYS';
```

role_name	grantee_name
SYS_MANAGE_ROLES_ROLE	USER1
SYS_MANAGE_ROLES_ROLE	USER2

USER1 and USER2 are the only designated global role administrators.

4.2.7 Managing Roles

Add and remove users, and system, object, and schema privileges to and from roles.

In this section:

Delete a user-defined role from the database as long as all dependent roles retain the minimum required number of administrator users with active passwords. If the minimum value is not maintained, the command fails.

4.2.7.1 Creating a User-Defined Role

Create a new user-defined role.

Prerequisites

- Requires the `MANAGE ROLES` system privilege.

Context

When creating a user-defined role, you can designate administrators for the role or add them later (default). See Adding or Removing Role Administrators from a Role [page 108]. If you do not specify any administrators, then global role administrators (any user or role granted the `MANAGE ROLES` system privilege) can manage the role.

When creating a role, the `ADMIN` clause lets you control whether the designated role administrators are also members of the role.

Clause	Behavior
`WITH ADMIN ONLY OPTION`	The designated role administrators can only manage the role. They do not inherit any of the roles or privileges of the role.
`WITH ADMIN OPTION`	The designated role administrators can manage the role and as members of the role, inherit all roles or privileges of the role.

If role administrators are specified at role creation, then global role administrators cannot administer the role unless the `SYS_MANAGE_ROLES_ROLE` role is explicitly granted to the role. For this reason, SAP recommends that you always include `SYS_MANAGE_ROLES_ROLE` in the list of role administrators when designating role administrators during role creation.

Procedure

To create a role, execute one of the following statements:

Administrative Type	Syntax

Allow only global role administrators to manage the role.	`CREATE ROLE <role_name>`
	
	Example:
	
	`CREATE ROLE ROLE_1;`

Allow specific role administrators to administer the role but do not make them members of the role.	`CREATE ROLE <role_name> WITH ADMIN ONLY <admin_name>[,...]`
	
	Example:
	
	`CREATE ROLE ROLE_2 WITH ADMIN ONLY USER1, USER2;`

Allow specific role administrators to administer the role and be members of the role.	`CREATE ROLE <role_name> WITH ADMIN <admin_name>[,...]`
	
	Example:
	
	`CREATE ROLE ROLE_3 WITH ADMIN USER1, USER2;`

Allow specific role administrators and global role administrators to manage the role, but not become members of the role.	`CREATE ROLE <role_name> WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, <admin_name>[,...]`
	
	Example:
	
	`CREATE ROLE ROLE4 WITH ADMIN ONLY USER1, USER2, SYS_MANAGE_ROLES_ROLE;`
	
	CREATE ROLE Statement

Related Information

CREATE ROLE Statement

4.2.7.2 Adding or Removing a Role from Users or Roles

Grant and revoke a role to and from users or role.

Prerequisites

- Requires administrative rights over the role being granted.
- If the grantee is another role, also requires administrative rights over the grantee role.

Context

Membership in a role can be granted to users and other roles with or without administrative rights.

- `WITH NO ADMIN OPTION` (default) - Members of the role inherit all underlying privileges of the granted role, but cannot administer the granted role.
- `WITH ADMIN ONLY OPTION` - Members of the role can administer the granted role but do not inherit any privileges or roles of the granted role.
- `WITH ADMIN OPTION` - Members of the role inherit all underlying privileges and roles of the granted role and can administer the granted role.
- `WITH NO SYSTEM PRIVILEGE INHERITANCE` - If the role being granted is a user-extended role, the underlying system privileges of the role are inherited only by the extended user, not the members of the extended role.

> ⚠ Restriction
>
> When regranting a role, if the grantee already has membership in the role with rights higher than the current grant, the grantee retains the original rights and inherits the higher rights. But if the regrant is for lesser rights than the original grant, the higher rights are `NOT` reduced for the grantee.

A role's membership in another role can be entirely revoked, including the ability to administer the role (if granted) along with any privileges and roles inherited from role membership. If granted both membership in a role along with administrative rights on the role, membership only can be removed while retaining the ability to manage the role. Finally, if granted both membership and administrative rights, only administrative rights can be removed while retaining the grantee's membership in the role.

A minimum number of role or global role administrators (as defined by the `MIN_ROLE_ADMINS` database option) with a login password must exist for each role at all times. If the member of the role is an administrator of the role and their removal violates the minimum requirement, the removal fails.

Procedure

To add or remove a user or role to or from a role, execute one of the following statements:

Grant Type	Statement

Members of the role inherit the underlying roles and privileges of the granted role, but cannot administer the granted role.	`GRANT ROLE <role-name> TO { <role-name> | <user-name> }[,...] [ WITH NO ADMIN OPTION ]`
	Example:
	`GRANT ROLE ROLE_2 TO ROLE_1, USER1 WITH NO ADMIN OPTION;`

Members of the role can administer the granted role, but do not inherit the underlying roles and privileges of the granted role.	`GRANT ROLE <role-name> TO { <role-name> | <user-name> }[,...] WITH ADMIN ONLY OPTION`
	Example:
	`GRANT ROLE ROLE_2 TO ROLE_1, USER1 WITH ADMIN ONLY OPTION;`

Members of the role inherit the underlying roles and privileges of the granted role and can administer the granted role.	`GRANT ROLE <role-name> TO { <role-name>| <user-name> }[,...] WITH ADMIN OPTION`
	Example:
	`GRANT ROLE ROLE_2 TO ROLE_1, USER1 WITH ADMIN OPTION;`

Members of the role lose the underlying roles and privileges of the revoked role along with the ability to administer the role.	`REVOKE ROLE <role_name> FROM <role_name>[,...]`
	Example:
	`REVOKE ROLE ROLE_2 FROM ROLE_1;`

Members of the role lose the ability to administer the role, but retain inheritance of the underlying roles and privileges of the role, if granted.	`REVOKE ADMIN OPTION FOR ROLE <role_name> FROM <role_name>[,...]`
	Example:
	`REVOKE ADMIN OPTION FOR ROLE ROLE_2 FROM ROLE_1;`

Members of the role loses the underlying roles and privileges of the role, but retain the ability to administer the role, if granted.	`REVOKE EXERCISE OPTION FOR ROLE <role_name> FROM <role_name>[,...]`
	Example:
	`REVOKE EXERCISE OPTION FOR ROLE ROLE_2 FROM ROLE_1;`

Related Information

GRANT ROLE Statement
REVOKE ROLE Statement

4.2.7.3 Adding or Removing System Privileges from a Role

Grant and revoke system privileges to and from roles.

Prerequisites

- Requires administrative privilege over the system privilege being granted.
- Requires administrative privilege of the role.

Context

System privileges can be granted with or without administrative rights.

- `NO ADMIN OPTION` (Default) - Members of the role inherit the role's underlying privileges, but cannot administer the role.
- `WITH ADMIN ONLY OPTION` - Members of the role can administer the role, but do not inherit the role's underlying privileges.
- `WITH ADMIN OPTION OPTION` - Members of the role can administer the role and inherit the role's underlying privileges.

> ⚠ Restriction
>
> When regranting a system privilege, if the role already has the system privilege granted with rights higher than the current grant, the role retains the original rights and inherits the higher rights. But if the regrant is for lesser rights than the original grant, the higher rights are `NOT` reduced for the role.

When revoking a system privilege from a role, you can remove the system privilege entirely from the role or remove only the ability to administrative the privilege from the role. If the privilege was originally granted to the role using the `WITH ADMIN OPTION ONLY OPTION` clause, you cannot remove only the ability to use the privilege from the role, leaving the ability to administer the privilege.

Procedure

To add or remove a system privilege to or from a role, execute one of the following statements:

Grant Type	Statement

Role can use the privilege, but cannot administer the privilege.	`GRANT <system_privilege_name> TO <role_name>[,...] [ WITH NO ADMIN OPTION ]`
	Example:
	`GRANT PRIV1 TO ROLE_1 WITH NO ADMIN;`

Role can administer the privilege, but cannot use the privilege.	`GRANT <system_privilege_name> TO <role_name>[,...] WITH ADMIN ONLY OPTION`
	Example:
	`GRANT PRIV1 TO ROLE_1 WITH ADMIN ONLY OPTION;`

Role can use the privilege and administer it.	`GRANT <system_privilege_name> TO <role_name>[,...] WITH ADMIN OPTION`
	Example:
	`GRANT PRIV1 TO ROLE_1 WITH ADMIN OPTION;`

Role loses the ability to use the privilege or administer it.	`REVOKE <system_privilege_name> FROM <role_name>[,...]`
	Example:
	`REVOKE PRIV1 FROM ROLE_1;`

Role loses the ability to administer the privilege, but retains the ability to use the privilege.	`REVOKE ADMIN OPTION FOR <system_privilege_name> FROM <role_name>[,...]`
	Example:
	`REVOKE ADMIN OPTION FOR PRIV1 FROM ROLE_1;`

Related Information

GRANT System Privilege Statement
REVOKE System Privilege Statement

4.2.7.4 Adding and Removing an Object Privilege from Roles

Grant and revoke an object privilege to and from roles.

Prerequisites

- Requires administrative rights over the role being granted.
- Also requires one of:
  - You own the object the privilege is begin granted on.
  - `MANAGE ANY OBJECT PRIVILEGE` system privilege.
  - Administrative rights over the object privilege being granted.

Context

Object privileges can be granted to a role with or without the administrative rights.

When granted using the `WITH GRANT OPTION` clause, members of the role inherit the privilege and can administer the privilege. Otherwise, members of the role inherit the privilege but cannot administer the privilege. If not specified, the privilege is granted without administrative rights.

Once a privilege is granted with administrative rights, you cannot revoke only the ability to administer from the privilege. Instead, revoke the privilege and then regrant the privilege without the `WITH GRANT OPTION` clause.

Procedure

To add or remove an object privilege to or from a role, execute one of the following statements:

Grant Type	Statement
Members of the role inherit the object or schema privileges, but cannot administer the privileges.	1. `GRANT <object-privilege-name> ON <object-name> TO <role-name>[,...]` 2. `GRANT <schema-privilege-name> ON SCHEMA <schema-name> TO <role-name>[,...]` Example: 1. `GRANT SELECT ON T1 TO ROLE_1;` 2. `GRANT SELECT ON SCHEMA SCHEMA1 TO ROLE_1;`

Grant Type	Statement
Members of the role inherit the object or schema privilege and can administer the privilege.	1. `GRANT <object-privilege-name> ON <object-name> TO <role-name>[,...] WITH GRANT OPTION` 2. `GRANT <schema-privilege-name> ON <schema-name> TO <role-name>[,...] WITH GRANT OPTION` Example: 1. `GRANT SELECT ON T1 TO ROLE_1 WITH GRANT OPTION;` 2. `GRANT SELECT ON SCHEMA SCHEAM1 TO ROLE_1 WITH GRANT OPTION;`
Members of the role lose the object or schema privilege along with the ability to administer the privilege.	1. `REVOKE <object_name> FROM <role_name>[,...]` 2. `REVOKE <schema-name> FROM <role_name>[,...]` Example: 1. `REVOKE SELECT ON T1 FROM ROLE_1;` 2. `REVOKE SELECT ON SCHEMA SCHEMA1 FROM ROLE_1;`

Related Information

GRANT Object-Level Privilege Statement
REVOKE Object-Level Privilege Statement

4.2.7.5 Adding Role and Global Role Administrators When Creating a Role

Specify role administrators or global role adminsitrators when creating a new role.

Prerequisites

MANAGE ROLES system privilege.

Context

If you specify at least one role administrator when you create a role, global role administrators cannot manage the role unless explicitly specified.

For this reason, SAP strongly recommends that you consider always adding the global role administrator to the list of role administrators.

Procedure

To add role administrators during the creation process, execute one of these statements:

Create Type	Statement

Role administrators have administrative rights only on the role; members of the role do not inherit the privileges of the role	`CREATE ROLE <role_name>`
	`    WITH ADMIN ONLY <admin_name [,...]>`
	Example:
	`CREATE ROLE Role1 WITH ADMIN ONLY user1, user2;`

Role and global role administrators have administrative rights only on the role; members of the role do not inherit the privileges of the role	`CREATE ROLE <role_name> WITH ADMIN ONLY`
	`    SYS_MANAGE_ROLES_ROLE, <admin_name [,...]>`
	Example:
	`CREATE ROLE Role1 WITH ADMIN ONLY`
	`   SYS_MANAGE_ROLES_ROLE, user1, user2;`

Role administrators have administrative rights on the role along with role membership; members of the role inherit the privileges of the role. To add global role administrators to the role requires a separate grant statement.[**]	`CREATE ROLE <role_name>`
	`    WITH ADMIN <admin_name [,...]>`
	`GRANT ROLE <role_name> TO SYS_MANAGE_ROLES_ROLE`
	`    WITH ADMIN ONLY OPTION`
	Example:
	`CREATE ROLE Role1 WITH ADMIN user1, user2;`
	`GRANT ROLE Role1 TO SYS_MANAGE_ROLES_ROLE`
	`    WITH ADMIN ONLY OPTION;`

[**]Global role administrators cannot be granted membership in a role. Therefore, you cannot include `SYS_MANAGE_ROLES_ROLE` in the administrators list when you create a role with role administrators granted membership in the role (`WITH ADMIN OPTION`).

Related Information

CREATE ROLE Statement
GRANT ROLE Statement

4.2.7.6 Listing the System Privileges and Roles Granted to a Role

Display the roles and system privileges granted directly and indirectly to a role.

Prerequisites

Requires one of the following:

- You are a member of the role granted with administrative rights.
- `MANAGE ROLES` system privilege

Context

The `sp_displayroles` stored procedure returns the system privileges and roles granted to a specified role. You can tailor the output to show only direct grants or both direct and inherited grants. The output also indicates whether administrative rights are granted for each role or system privilege.

> **Note**
>
> If no role name is specified, the procedure returns only the direct grants of the connected user, not a role.

Procedure

To display both direct and inherited grants for a role, execute:

```
CALL sp_displayroles(<role-name>,'EXPAND_DOWN');
```

Sample Output:

```
CALL sp_displayroles('<role_1>','EXPAND_DOWN');
```

	role_name	parent_role_name	grant_type	role_level
1	ROLE_2	NULL	NO ADMIN	1
2	PUBLIC	NULL	NO ADMIN	1
3	ALTER ANY MATERIALIZED VIEW	NULL	ADMIN	1
4	ROLE_3	ROLE_2	NO ADMIN	2
5	ALTER ANY MATERIALIZED VIEW	ROLE_2	NO ADMIN	2
6	ALTER ANY SEQUENCE	ROLE_3	ADMIN	3

Where:

- A `parent_role_name` of `NULL` and a `<role_level>` of 1 indicate a direct grant.
- A `parent_role_name` with a role name and a `role_level` greater than 1 indicates an inheritance from the parent role.
- `grant_type` indicates whether the grant was with or without administrative privilege.

Summary of output:

Row 1

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that members of the `ROLE_1` role are directly granted membership in the `ROLE_2` role. The `<grant_type>` of `ADMIN` means that members of `ROLE_1` have administrative rights on the `ROLE_2` role.

Row 2

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that members of the `ROLE_1` role are directly grant membership in the `PUBLIC` role . The `<grant_type>` value of `NO ADMIN` means that members of `ROLE_1` don't have administrative rights on the `PUBLIC`role.

Row 3

A `<parent_role_name>` value of `NULL` and a `<role_level>` value of 1 means that members of the `ROLE_1` role are directly grant the `ALTER ANY MATERIALIZED VIEW` system privilege. The `<grant_type>` value of `ADMIN` means that members of `ROLE_1` have administrative rights on the system privilege.

Row 4

A `<parent_role_name>` value of `ROLE_2` and a `<role_level>` value of 2 means that members of the `ROLE_1` role inherit membership in the `ROLE_3` role through their membership in the `ROLE_2` role. The `<grant_type>` of `NO ADMIN` means that members of `ROLE_1` don't have administrative rights on the `ROLE_3` role.

Row 5

A `<parent_role_name>` value of `ROLE_2` and a `<role_level>` value of 2 means that members of the `ROLE_1` role inherit the grant of the `ALTER ANY VIEW` system privilege through their membership in `ROLE_2`. The `<grant_type>` of `NO ADMIN` means that members of `ROLE_1` don't have administrative rights on the `ALTER ANY VIEW` system privilege.

Row 6

A <parent_role_name> value of ROLE_3 and a row_level value of 3 means that members of the ROLE_1 role inherit the grant of the ALTER ANY SEQUENCE system privilege through their membership in ROLE_3 which is inherited from their membership in ROLE_2. The <grant_type> of ADMIN means that members of ROLE_1 have administrative privilege on the ALTER ANY SEQUENCE system privilege, even if they don't have administrative rights on the ROLE_3 role itself.

Related Information

sp_displayroles System Procedure

4.2.7.7 Listing Object Privileges Granted to a Role

List the object privileges granted directly or indirectly by a role. Object privileges include schema and personal security environment (PSE) specific privileges.

Prerequisites

- You are listing object privileges granted to self.
- MANAGE ANY OBJECT PRIVILEGE system privilege

Context

The results for sp_objectpermission include inherited object privileges as well as privileges explicitly granted to the user. Use the grantee column to learn where the object privilege is inherited from. The grantor column tells you who performed the actual granting. The grantable column tells you whether the user has administrative rights on the privilege.

Procedure

To display the object level privileges granted to a user, execute:

```
SELECT * FROM sp_objectpermission( '<role-name>' ) WHERE GRANTOR <> 'SYS';
```

Sample Output:

```
SELECT * FROM sp_objectpermission( 'role1' ) WHERE GRANTOR <> 'SYS';
```

	grantor	grantee	object_name	owner	object_type	column_name	permission	grantable
1	USER1	role1	table1	user1	COLUMN	col	SELECT	N
2	USER1	role1	pse1	user1	PSE	NULL	ALTER	Y
3	USER1	role2	my_schema1	user1	SCHEMA	NULL	SELECT	N
4	USER1	role2	proc1	user1	PROCEDURE	NULL	EXECUTE	y

Summary of Output:

Row 1 The `SELECT` privilege on column `col1` of table `table1` has been granted `role1` with no administrative rights on the column.

Row 2 The `ALTER` privilege on PSE `pse1` has been granted to `role1` with administrative rights.

As a member of `role2`, `role1` inherits the following object privileges from `role2`:

Row 3 The `SELECT` privilege on schema `my_schema1` has been inherited by `role1` with no administrative rights.

Row 4 The `EXECUTE` privilege on procedure `proc1` has been inherited by `role1`with administrative rights.

Related Information

sp_objectpermission System Procedure

4.2.7.8 Listing Members of a Role

Display all users and roles granted membership in a role.

Prerequisites

No additional privileges are required.

Procedure

To display all members, execute:

```
SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
   WHERE role_name ='<role-name>'
```

Where:

grant_type	How role granted
5	Granted with `WITH NO ADMIN OPTION` clause (role only)
6	Granted with `WITH ADMIN ONLY OPTION` clause
7	Granted with `WITH ADMIN OPTION` clause

Example

This example lists all members of role `ROLE1`:

```
SELECT role_name, grantee_name, grant_type, grantor FROM SYSROLEGRANTS
   WHERE role_name ='ROLE1';
```

role_name	grantee_name	grant_type	grantor
ROLE1	ROLE2	5	USER1
ROLE1	SYS_MANAGE_ROLES_ROLE	6	USER1
ROLE1	USER1	6	USER1
ROLE1	USER2	7	USER1
ROLE1	USER3	5	USER1

`ROLE2` and `USER3` are granted role membership only (5). `SYS_MANAGE_ROLES_ROLE` and `USER1` are granted role administrative rights only (6). `USER2` is granted role membership and administrative rights (7).

4.2.7.9 Deleting a Role

Delete a user-defined role from the database as long as all dependent roles retain the minimum required number of administrator users with active passwords. If the minimum value is not maintained, the command fails.

Prerequisites

Requires one of the following:

- Administrative privilege on the role.

Context

You cannot delete a user-extended role. You must first convert it back to act as a user (see Converting a User-Extended Role Back to a User [page 94]) and then delete the user. See Deleting a User [page 32].

You cannot drop system roles.

Procedure

To delete a user-defined role, execute one of the following statements:

Delete Condition	Statement

Role has no members.	`DROP ROLE <role_name>`
	Example:
	`DROP ROLE ROLE_1;`

Role has members.	`DROP ROLE <role_name> WITH REVOKE`
	Example:
	`DROP ROLE ROLE_2 WITH REVOKE;`

Related Information

DROP ROLE Statement

4.2.8 Referencing Objects Owned by User_Extended Roles

Views, procedures, and tables are more easily managed when they are owned by a user-extended role instead of a user.

To eliminate having to qualify the object name, make users who need access to a table, view, or stored procedure members of the role that owns the object.

For example, the table `Employees` is owned by the role `Personnel`, of which `Jeff` is a member. When `Jeff` wants to refer to the `Employees` table, he need only specify the name of the table in SQL statements, for example:

```
SELECT * FROM EMPLOYEES
```

However, when `John`, who is not a member of `Personnel`, wants to refer to the `Employees` table, he must use the qualified name of the table, for example:

```
SELECT * FROM PERSONNEL.EMPLOYEES
```

> **Note**
>
> Since ownership of database objects is associated with a single user ID, when the owner is a role, ownership of the table is not inherited by members of the role.

DO not grant system privileges to roles that own objects. Instead:

- Create distinct roles with specific system privileges granted
- Grant users who require the specific system privileges membership to the applicable role
- Grant each distinct role to the role that owns the object.

This allows for complete control of the tasks performed by each user. Maintain authorized tasks by granting and revoking membership in the applicable role associated with the object.

For example, the table `Sales` is owned by the `Sales1` role. Users `Mary`, `Bob`, `Joe`, `Laurel`, and `Sally` are granted membership to `Sales1`. Create `Task1_role` and granted it the system privileges necessary to complete a specific task. Grant `Task1_role` to `Mary` and `Bob`. Create `Task2_role`, grant it specific system privileges, and grant it to `Joe` and `Sally`. Finally, grant both `Task1_role` and `Task2_role` to `Sales1`. Though both roles are granted to `Sales1`, the underlying system privileges of `Task1_role` and `Task2_role` are not automatically inherited by the other members of `Sales1`. `Mary` and `Bob` can perform different tasks than `Joe` and `Sally`. Since `Laurel` has not been granted to either `Task1_role` or `Task2_role`, and no system privileges have been granted directly to `Sales1`, `Laurel` can perform no privileged tasks on the `Sales` table. This configuration allows you to maintain and control the tasks that can be performed by each user.

4.2.9 Determining the Roles and Privileges Granted to a User

The `sp_has_role` stored function returns an integer value that indicates whether the invoker of the procedure has been granted the specified system privilege or user-defined role.

No system privileges are required to execute the function. When used for permission checking within user-defined stored procedures, this function can display an error message when a user fails a permission check.

1 indicates the system privilege or user-defined role is granted to the invoking user.

0 or Permission denied: you do not have permission to execute this command/procedure indicates the system privilege or user-defined role is not granted to the invoking user. The error message replaces the value 0 when the throw_error argument is set to 1.

-1 indicates the system privilege or user-defined role specified does not exist. No error message appears, even if the throw_error argument is set to 1.

Related Information

SP_HAS_ROLE Function [System]

4.3 Restrictions when Regranting Privileges and Roles

When regranting a system privilege or role, the regrant can expand the grantee's existing rights, but cannot reduce or diminish them.

If a system privilege or role is regranted with higher rights than the original grant, then the grantee retains the original rights and inherits the higher rights. But if the regrant is for lesser rights than the original grant, then the higher rights are `NOT` diminshed by the regrant. The grantee inherits any new right from the regrant while retaining any higher rights from the original grant.

This restriction applies whether the item being granted is a privilege or a role, and the grantee is a user or role. As a result, depending on how the original grant was made, regranting a privilege or role may produce unexpected results.

These restrictions do not apply when regranting object or schema privileges.

Privilege Regrants:

A privilege can be granted to a grantee (user or role) in one of three ways:

- `NO ADMIN` - Grantee can use the privilege (execute tasks requiring the privilege), but cannot administer the privilege (grant and revoke the privilege to and from other users and roles).
- `ADMIN ONLY` - Grantee can administer the privilege but cannot use the privilege.
- `ADMIN` - Grantee can both use and administer the privilege.

Scenario 1 - Regrant higher rights:

`USER1` is initially granted a privilege without administrative rights. This means `USER1` could perform tasks that require the privilege, but cannot administer the privilege. `USER1` is then regranted the privilege with administrative rights. The expected result of the regrant is for `USER1` to continue to be able to use the privilege and now be able to administer the privilege.

Both the initial and subsequent regrant allows `USER1` to use the privilege, so `USER1` retains this right. Since the regrant allows the user to administer the privilege, `USER1` inherits this higher right. `USER1` can both use and administer the privilege. The actual result of the regrant matches the expected result.

Scenario 2 - Regrant lower rights:

`USER1` is initially granted a privilege with administrative rights. This means `USER1` can both use and administer the privilege. `USER1` is then regranted the privilege without administrative rights. The expected result of the regrant is for `USER1` to continue to be able to use the privilege, but no longer be able to administer the privilege.

Both the initial and subsequent regrant allows `USER1` to use the privilege, so `USER1` retains this right. The initial grant allowed `USER1` to administer the privilege while the subsequent regrant does not. Since a regrant cannot diminish a grantee's existing rights, `USER1` retains the ability to administer the privilege. Instead of only being able to use the privilege (expected), `USER1` can both use and administer the privilege (actual). The actual result of the regrant does not match the expected result.

Scenario 3: - Changed Rights Regrant

USER1 is initially granted a privilege with administrative rights only. This means USER1 can administer the privilege, but cannot use it. USER1 is then regranted the privilege without administrative rights. The expected result of the regrant is for USER1 to be able to use the privilege, but no longer administer it.

The initial grant allowed USER1 to administer the privilege while the subsequent regrant doesn't. Since the regrant cannot diminish this right, USER1 retains the ability to administer the privilege. The regrant also grants the ability to use the privilege, a higher right than the original grant, so USER1 inherits this higher right. Instead of only being able to use the privilege (expected), USER1 can both use and administer the privilege (actual). The actual result of the regrant does not match the expected result.

To prevent unexpected results of a regrant, best practice is to revoke the privilege in its entirety before regranting it.

Role Regrants

Membership in a role can be granted to a grantee (user or role) in one of the three ways:

- NO ADMIN - Grantee inherits the underlying privileges of the role, but cannot administer the role (grant, revoke, or drop the role).
- ADMIN ONLY - Grantee can administer the role, but does not inherit the underlying privileges of the role.
- ADMIN - Grantee inherits the underlying privileges of the role and can administer the role.

Scenario 1 - Higher Rights Regrant:

USER1 is initially granted membership in ROLE1 with no administrative rights. This means that as a member of ROLE1, USER1 inherits the underlying privileges of the role, but cannot administer the role. USER1 is then regranted membership in ROLE1 with administrative rights. The expected result of the regrant is that USER1 both inherits the underlying privileges and can administer the role.

Both the initial and subsequent regrant allows USER1 to inherit the role's underlying privileges, so USER1 retains this right. Since the regrant allows the user to administer the role, USER1 inherits this higher right. USER1 both inherits the underlying privileges of the role and can administer the role. The actual result of the regrant matches the expected result.

Scenario 2 - Regrant lower rights:

USER1 is initially granted membership in ROLE1 with administrative rights. This means that as a member of ROLE1, USER1 inherits the underlying privileges of the role and can administer the role. USER1 is then regranted membership in the role without administrative rights. The expected result of the regrant is for USER1 to be continue to inherit the underlying privileges of the role, but no longer be able to administer the role.

Both the initial and subsequent regrant allows USER1 to inherit the role's underlying privileges, so USER1 retains this right. But since the initial grant allowed USER1 to administer the role while the subsequent regrant does not, and a regrant cannot diminish a grantee's existing rights, USER1 retains the ability to administer the role. Instead of only inheriting the role's underlying privileges, (expected), USER1 now inherits the underlying privileges of the role while retaining the ability to administer the role (actual). The actual result of the regrant does not match the expected result.

Scenario 3: - Changed Rights Regrant

USER1 is initially granted membership in ROLE1 with administrative rights only. This means that as a member of ROLE1, USER1 can administer the role, but does not inherit the underlying privileges of the role. USER1 is

then regranted membership in the role without administrative rights. The expected result of the regrant is for USER1 to inherit the underlying privileges of the role, but no longer be able to administer the role.

The initial grant allowed USER1 to administer the role while the subsequent regrant doesn't. Since the regrant cannot diminish this right, USER1 retains the ability to administer the role. The regrant also grants the ability to inherit the underlying privileges of the role, a higher right than the original grant, so USER1 inherits this new right. Instead of only inheriting the underlying privileges of the role (expected), USER1 now inherits the underlying privileges of the role while retaining the ability to administer the role (actual). The actual result of the regrant does not match the expected result.

5 Authentication

SAP IQ supports several authentication mechanisms.

Mechanism	Description
SAP IQ user name and password	Users accessing SAP IQ authenticate themselves by entering their database user name and their local password.
LDAP	Use a password stored in an LDAP directory server to authenticate users. Users access SAP IQ directly from database clients. See LDAP User Authentication with SAP IQ [page 195].

In this section:

5.1 Login Policies

A login policy defines the rules that SAP IQ follows to establish user connections. Each login policy is associated with a set of options called login policy options.

Login management commands that you execute on any multiplex server are automatically propagated to all servers in the multiplex. For best performance, execute these commands, or any DDL, on the coordinator.

Each new database is created with a default login policy, called ROOT. When you create a user account, if you don't specify a login policy, then the user is assigned the ROOT login policy. The ROOT login policy can be modified, but not deleted.

Each user can be associated with only one login policy. You can create as many user-defined login policies with as many combinations of login policy and LDAP login policy options as needed. A new user-defined login policy starts with the default values as defined in the ROOT policy. Add only those policy options that require different values from the default value to the user-defined policy. To restore an option value in a user-defined login policy back to the default value, set the value to DEFAULT in the policy.

Use a login policy to permanently lock a user account. This prevents a user from connecting without having to delete the user account. Create a login policy with the LOCK option set to ON. Then, assign the login policy to any user to be personalty locked. To unlock a user account, assign a login policy with the LOCK option set to OFF or DEFAULT to the user.

To control how many failed login attempts are allowed and temporarily lock the account once the maximum attempts is exceeded, create a login policy with the MAX_FAILED_LOGIN_ATTEMPTS option. Once an

account is temporarily locked, to define a wait period before the user than try logging in again, add the AUTO_UNLOCK_TIME option to the same login policy. If the AUTO_UNLOCK_TIME option is not added to the policy, the user account remains temporarily locked until it is manually unlocked using the ALTER USER statement. The default value in ROOT for the MAX_FAILED_LOGIN_ATTEMPTS is unlimited. SAP strongly recommends that a login policy with a fixed failed attempts limit be implanted.

In this section:

5.1.1 Login Policy Options

Available options for ROOT (default) and user-defined login policies.

When applying a policy option, the system verifies that the defined value falls within the predefined range. If it doesn't, then the system overrides the defined value with the upper limit of the option.

For example, the `PASSWORD_LIFE_TIME` option has a valid range of 0 - 2147483647. If you set the option to 3000000000, then when the option is referenced, the value 3000000000 is automatically replaced with the highest allowed value, which is 2147483647.

Option	Description	Properties
AUTO_UN-LOCK_TIME	The time period after which locked accounts that are not granted the MANAGE ANY USER system privilege are automatically unlocked. You can define this option in any login policy, including the root login policy.	• Values – 0 – UNLIMITED • Default – UNLIMITED • Applies to all users who are not granted the MANAGE ANY USER system privilege

Option	Description	Properties
CHANGE_PASSWORD_DUAL_CONTROL	Requires input from two users, each of whom is granted the CHANGE PASSWORD system privilege, to change the password of another user.	• Values – ON; OFF • Default – OFF • Applies to all users

DEFAULT_LOGICAL_SERVER	If the connection string specifies no logical server, the user connects to the DEFAULT_LOGICAL_SERVER option specified in the user's login policy.	• Values:
		  • Name of an existing user-defined logical server.
		  • AUTO – value of the default logical server in the root login policy.
		  • COORDINATOR – the current coordinator node.
		  • NONE – denies access to any multiplex server.
		  • OPEN – use alone or with the name of a user-defined logical server. Allows access to all multiplex nodes that are not members of any user-defined logical servers.
		  • SERVER – allows access to all of the multiplex nodes, subject to the semantics of the SERVER logical server.
		• Default – AUTO
		• Applies to all users. Requires MANAGE MULTIPLEX system privilege

LOCKED	If set ON, users cannot establish new connections. This setting temporarily denies access to login policy users. Logical server overrides for this option are not allowed.	• Values – ON; OFF • Default – OFF • Applies to all users except those with the MANAGE ANY USER system privilege
MAX_CONNECTIONS	The maximum number of concurrent connections allowed for a user. You can specify a per-logical-server setting for this option.	• Values – 0–2147483647 • Default – UNLIMITED • Applies to all users except those with the SERVER OPERATOR or DROP CONNECTION system privilege
MAX_DAYS_SINCE_LOGIN	The maximum number of days that can elapse between two successive logins by the same user.	• Values – 0–2147483647 • Default – UNLIMITED • Applies to all users except those with the MANAGE ANY USER system privilege
MAX_FAILED_LOGIN_ATTEMPTS	The maximum number of failed attempts, since the last successful attempt, to log in to the user account before the account is locked.	• Values – 0–2147483647 • Default – UNLIMITED • Applies to all users

Option	Description	Properties
MAX_NON_DBA_CONNECTIONS	The maximum number of concurrent connections that a user without SERVER OPERATOR or DROP CONNECTION system privileges can make. This option is supported only in the root login policy.	• Values – 0–2147483647 • Default – UNLIMITED • Applies to all users except those with the SERVER OPERATOR or DROP CONNECTION privilege
PAM_FAILOVER_TO_STD	Use standard authentication if PAM authentication is enabled but the PAM library is unavailable due to a system failure. Authentication failures returned by PAM do not fail over to standard authentication.	• Values – ON; OFF • Default – ON • Applies to all users
PAM_SERVICE_NAME	The PAM service name to use when authenticating. The service name identifies the rule set to be used by PAM during validation. If empty (the default), do not use PAM. The database server continues to function when PAM support is unavailable. See *Enabling PAM User Authentication* in *Administration: User Management and Security*.	
PASSWORD_EXPIRY_ON_NEXT_LOGIN	If set ON, the user's password expires at the next login. **Note** This functionality is not currently implemented when logging in to SAP IQ Cockpit. However, when logging in to SAP IQ outside of SAP IQ Cockpit (for example, using Interactive SQL), users are then prompted to enter a new password.	• Values – ON; OFF • Default – OFF • Applies to all users
PASSWORD_GRACE_TIME	The number of days before password expiration during which login is allowed but the default post_login procedure issues warnings.	• Values – 0–2147483647 • Default – 0 • Applies to all users.
PASSWORD_LIFE_TIME	The maximum number of days before a password must be changed.	• Values – 0–2147483647 • Default: UNLIMITED • Applies to all users
ROOT_AUTO_UNLOCK_TIME	The time period after which locked accounts that are granted the MANAGE ANY USER system privilege are automatically unlocked. You can define this option only in the root login policy.	• Values: 0 – UNLIMITED • Default: 15 • Applies to all users who are granted the MANAGE ANY USER system privilege.

5.1.2 LDAP Login Policy Options

Available login policy options for LDAP user authentication

Option	Description	Properties
LDAP_AUTO_FAIL-BACK_PERIOD	Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.	Values – 0–2147483647 Default – 15 minutes Applies to all users
LDAP_FAIL-OVER_TO_STD	Permits authentication with standard authentication when authentication with the LDAP server fails due to system resources, network outage, connection timeouts, or similar system failures. However, it does not permit an actual authentication failure returned from an LDAP server to fail over to standard authentication.	Values – ON; OFF Default – ON Applies to all users
LDAP_PRI-MARY_SERVER	Specifies the name of the primary LDAP server.	Values – N/A Default – none Applies to all users
LDAP_REFRESH_DN	Updates the ldap_refresh_dn value in the ISY-SLOGINPOLICYOPTION system table with the current time, stored in Coordinated Universal Time (UTC). Each time a user authenticates with LDAP, if the value of ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the value of user_dn in ISYSUSER, a search for a new user DN occurs. The user_dn value is then updated with the new user DN and the user_dn_changed_at value is again updated to the current time.	Values – NOW Initial value for ROOT policy – NULL Initial value for user-defined login policy – current time stored in UTC
LDAP_SECON-DARY_SERVER	Specifies the name of the secondary LDAP server.	Values – N/A Default – none Applies to all users

5.1.3 Creating a New Login Policy

Any options that are not explicitly set when creating a login policy inherit their values from the root login policy.

Prerequisites

- Requires the MANAGE ANY LOGIN POLICY system privilege.

Context

Login policy names are unique. An error appears if the login policy name already exists.

Procedure

Execute:

```
CREATE LOGIN POLICY <policy_name> {<login_policy_options>}
```

Example

This statement creates the `Test1` login policy with PASSWORD_LIVE_TIME option set to 60 days:

```
CREATE LOGIN POLICY Test1 password_life_time=60
```

Related Information

CREATE LOGIN POLICY Statement
Login Policy Options [page 137]
LDAP Login Policy Options [page 140]
Multiplex Login Policy Configuration

5.1.4 Modifying an Existing Login Policy

Modify options within an existing login policy.

Prerequisites

- Requires the MANAGE ANY LOGIN POLICY system privilege.

Procedure

1. (Optional) Review the policy options currently assigned to the login policy:

```
SELECT SYSLOGINPOLICY.login_policy_name,
SYSLOGINPOLICYOPTION.login_option_name,
SYSLOGINPOLICYOPTION.login_option_value
   FROM SYSLOGINPOLICYOPTION
   JOIN SYSLOGINPOLICY ON
SYSLOGINPOLICY.login_policy_id=SYSLOGINPOLICYOPTION.login_policy_id
   WHERE login_policy_name='<policy-name>';
```

2. Alter the options of an existing login policy by executing:

```
ALTER LOGIN POLICY <policy-name> <policy_option> [...]
```

Example

This statement alters the LOCKED and MAX_CONNECTIONS options on the `Test1` login policy:

```
ALTER LOGIN POLICY Test1 locked=on max_connections=5;
```

This example alters the value of `LOCKED` back to the default value of `OFF` in login policy `Test1`:

```
ALTER LOGIN POLICY Test1 LOCKED=DEFAULT;
```

Related Information

ALTER LOGIN POLICY Statement
Login Policy Options [page 137]
LDAP Login Policy Options [page 140]
Multiplex Login Policy Configuration

5.1.5 Deleting a Login Policy

You cannot delete the root login policy, or one that is currently assigned to a user.

Prerequisites

- Requires the `MANAGE ANY LOGIN POLICY` system privilege.

Procedure

1. Verify that no users are currently assigned the login policy to be dropped.

```
SELECT SYSUSER.user_name, sysloginpolicy.login_policy_name
FROM SYSLOGINPOLICY
JOIN SYSUSER on SYSUSER.login_policy_id=SYSLOGINPOLICY.login_policy_id
WHERE SYSLOGINPOLICY.login_policy_name='<policy-name>';
```

2. To drop the login policy, execute:

```
DROP LOGIN POLICY <policy-name>;
```

Related Information

DROP LOGIN POLICY Statement

5.1.6 Assigning a Login Policy to an Existing User

Assign a login policy to an existing SAP IQ user.

Prerequisites

- Requires the MANAGE ANY LOGIN POLICY system privilege.

Procedure

1. Execute:

```
ALTER USER <userID> LOGIN POLICY <policy_name>
```

2. Have the user log out and back in to apply the new login policy.

Related Information

ALTER USER Statement

5.1.7 Automatic Unlocking of User Accounts

A user account is automatically locked if the user exceeds the maximum failed login attempts limit (`MAX_FAILED_LOGIN_ATTEMPTS`) value defined in the login policy.

Once locked, the user account can be manually unlocked using the `ALTER USER ... RESET LOGIN POLICY` statement by a user granted the `MANAGE ANY USER` system privilege. However, if all users with the `MANAGE ANY USER` system privilege are themselves locked out due to failed login attempts, a potential lock-down of some or all the services of a database can occur.

Automatic unlocking is applicable only to locked accounts due to failed login attempts and not to accounts locked for any other reason. The locked status of a user is verified during login, and if the user has equaled or exceeded the specified automatic unlock period, the user is allowed to log in and the failed_login_attempts counter is reset to zero.

Based on the permissions granted a user, one of the two login policy options is verified at the time of unlocking.

A lock-down of some or all of the services of a database can occur if all administrative users with the `MANAGE ANY USER` system privilege are locked out of the database due to failed login attempts.

To prevent this scenario, ensure that there is at least one user with the `MANAGE ANY USER` system privilege who is assigned a login policy with `AUTO_UNLOCK_TIME` set to `UNLIMITED`. In all other login policies, `AUTO_UNLOCK_TIME` should be set to a reasonable value (for example, 60 for a one-hour delay before automatic unlock occurs).

Configuration of these values requires the `MANAGE ANY LOGIN POLICY` system privilege.

5.2 Passwords

A user can be granted the ability to manage other users' passwords. You can configure password management to require one or two users to complete a password change.

In this section:

Related Information

Resetting the DBA Password (Command Line) [page 19]

5.2.1 User ID and Password Restrictions and Considerations

A user must have a password to connect to the database.

User ID Restrictions

The user name is an identifier and must follow the rules for Identifiers.

User IDs cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons

Password Restrictions

A password can be an identifier or a string literal as the password. If you specify an identifier then you must follow the rules for Identifiers. If you specify a string literal, then there are fewer restrictions, but you must enclose the password in single quotations.

By default, passwords must be 6 bytes in length. For single-byte character sets, this value is the same as the number of characters. To change this requirement, set the `min_password_length` database option.

A password must contain:

- at least one number character (0-9)
- at least one upper case Latin letter character (A-Z)
- at least one lower case Latin letter character (a-z)

Passwords are case-sensitive and they cannot:

- begin with white space, single quotes, or double quotes
- end with white space
- contain semicolons
- be longer than 255 bytes in length
- be fewer than 6 characters in length
- contain the user's ID (user name)
- be the re-use of an old password for that user

Additional rules for passwords can be set in a user's login policy.

When passwords are created or changed, they are converted to UTF-8, encrypted using a SHA256 hash, and then stored in the database. If the database is unloaded and reloaded into a database with a different character set, then existing passwords continue to work. If the database server cannot convert from the client's character set to UTF-8, then use 7-bit ASCII characters for the password as other characters may not work correctly.

Passwords are encrypted before they are sent from the client to the server.

Case-Sensitivity of Passwords

Case-sensitivity of passwords is treated differently from other identifiers.

All passwords in newly created databases are case-sensitive, regardless of the case-sensitivity of the database.

When you rebuild an existing database, SAP IQ determines the case-sensitivity of the password as follows:

- If the password was originally entered in a case-insensitive database, the password remains case-insensitive.
- If the password was originally entered in a case-sensitive database, uppercase and mixed-case passwords remain case-sensitive. If the password was entered in all lowercase, then the password becomes case-insensitive.
- Changes to both existing passwords and new passwords are case-sensitive.

Login Policies

Create login policies to ensure optimal security.

- Use the min_password_length Option to require long user passwords.
- Limit failed login attempts.
- Automatically lock users if no login attempts occur in a specified number of days.
- Specify password lifetime.
- Use theverify_password_function option to require complex passwords. This option can also be used to prevent password reuse.

For a full list of available login policy options, see Login Policy Options [page 137]

5.2.2 Increase Password Security

Passwords are an important part of any database security system. There are several options for increasing password security.

Implement a Login Policy control the frequency of password changes, to specify the number of login attempts allowed before an account is locked, or to force password expiration. See Login Policies [page 136].

Implement a Minimum Password Length by default, passwords can be any length. For greater security, you can enforce a minimum length requirement on all new passwords to disallow short (and therefore easily guessed) passwords. The recommended minimum length is 6. See min_password_length Option.

Implement Password Rules implement advanced password rules that include requiring certain types of characters in the password, disallowing password reuse, and expiring passwords. Validation of the rules occurs when a new user ID is created or a password is changed. See VERIFY_PASSWORD_FUNCTION_V2 Option.

5.2.3 Granting the CHANGE PASSWORD System Privilege to a User

Allow a user to manage the password of other users.

Prerequisites

- Requires the CHANGE PASSWORD system privilege granted with administrative rights.
- Each target user specified (target_users_list) must be an existing user or user-extended role with a login password.
- Each target role specified (target_roles_list) must be an existing user-extended or user-defined role.

Context

You can grant a user the ability to change the password of any user in the database (ANY), only specific users (`<target_users_list>`), or members of specific roles (ANY WITH ROLES `<target_roles_list>`). Administrative rights to the CHANGE PASSWORD system privilege can be granted only when using the ANY clause.

If no clause is specified, the default is ANY, WITH NO ADMIN OPTION.

When regranting the CHANGE PASSWORD system privilege, the effect of the grant is cumulative. For example, if you grant `User1` the privilege limited to `User2` and `User3`, and then regrant the privilege limited to `Role1`, `User1` can manage the password of `User2`, `User3`, and any member of `Role1`.

If you grant the CHANGE PASSWORD system privilege to a user with fewer rights than currently granted, the higher rights are retained. For example, if the privilege is granted using the ANY clause and then regranted using the `<target_users_list>` clause, the user retains the rights of the ANY clause.

Procedure

To grant the CHANGE PASSWORD system privilege, execute one of these statements:

Grant Type	Statement
Grant to any database user, with full administrative rights	`GRANT CHANGE PASSWORD (ANY) TO <user_ID> WITH ADMIN OPTION`
Grant to any database user, with administrative rights only	`GRANT CHANGE PASSWORD (ANY) TO <user_ID> WITH ADMIN ONLY OPTION`

Grant Type	Statement
Grant to any database user, with no administrative rights	`GRANT CHANGE PASSWORD (ANY) TO <user_ID>` `    WITH NO ADMIN OPTION`

Grant to specified users, with no administrative rights	`GRANT CHANGE PASSWORD`
	`(<target_users_list>)`
	`    TO <user_ID> WITH NO ADMIN OPTION`

Grant to any member of specified roles, with no administrative rights	`GRANT CHANGE PASSWORD (ANY WITH ROLES`
	`<target_roles_list>)`
	`    TO <user_ID> WITH NO ADMIN OPTION`

Grant to specified users, or any member of specified roles, with no administrative rights	`GRANT CHANGE PASSWORD`
	`<target_users_list>),`
	`    (ANY WITH ROLES <target_roles_list>)`
	`TO <user_ID>`
	`    WITH NO ADMIN OPTION`

Example

This statement grants `Sam` the ability to change the password of any database user:

```
GRANT CHANGE PASSWORD (ANY) TO Sam
or
GRANT CHANGE PASSWORD TO Sam
```

This statement grants `Sally` and `Bob` the ability to change the password for `Jane`, `<Joe>`, and `Laurel` only:

```
GRANT CHANGE PASSWORD (Jane, Joe, Laurel) TO Sally, Bob
```

This statement grants `Mary` the ability to change the password of any member of the `Sales1` role:

```
GRANT CHANGE PASSWORD (ANY WITH ROLES Sales1) TO Mary
```

This statement grants `Sarah` the ability to change the password of `Joe` or `Sue`, or any member of the `Sales2` role:

```
GRANT CHANGE PASSWORD (Joe, Sue), (ANY WITH ROLES Sales2) TO Sarah
```

This statement grants `Joan` the ability to change the password of any member of the `Marketing1` or `Marketing2` roles:

```
GRANT CHANGE PASSWORD (ANY WITH ROLES Marketing1, Marketing2) TO Joan
```

Related Information

GRANT CHANGE PASSWORD Privilege Statement

5.2.4 Revoking the CHANGE PASSWORD System Privilege from a User

Remove the ability of a user to manage passwords and administer the system privilege.

Prerequisites

- Requires the `CHANGE PASSWORD` system privilege granted with administrative rights.

Context

You can grant the CHANGE PASSWORD system privilege to a user multiple times, using different clauses. For example, `User1` is granted the CHANGE PASSWORD system privilege once using the ANY clause and again with the `<target_users_list>` clause. In cases of multiple grants, the same form of the clause used for the `GRANT` statement must be used to revoke it.

Continuing with the example, if the system privilege is revoked from `User1` using the ANY clause, the grant with the `<target_users_list>` clause remains in effect. The net effect is that `User1` is now limited to managing the passwords of users on the `<target_users_list>`. Alternately, if the system privilege is revoked from `User1` using the `<target_users_list>` clause, the grant with the ANY clause remains in effect. The net effect in this scenario is that `User1` can continue to manage the passwords of any user in the database.

Procedure

To revoke the CHANGE PASSWORD system privilege, execute one of these statements:

Revoke Type	Description
Revoke administrative rights only from the system privilege	`REVOKE ADMIN OPTION FOR CHANGE PASSWORD ( ANY ) FROM <user_ID [,...]>`
Revoke the system privilege, including administrative rights	`REVOKE CHANGE PASSWORD FROM <user_ID [,...]>`

Revoke Type	Description
Revoke the system privilege from specified users	`REVOKE CHANGE PASSWORD ( <target_users_list> ) FROM <user_ID [,...]>`
Revoke the system privilege from specified roles	`REVOKE CHANGE PASSWORD ( ANY WITH ROLES <target_roles_list> ) FROM <user_ID [,...]>`

Example

Both these statements remove the ability of `Sam` to change the password of any database user:

```
REVOKE CHANGE PASSWORD (ANY) FROM Sam
or
GRANT CHANGE PASSWORD TO Sam
```

Assuming that `Frank` was granted the CHANGE PASSWORD system privilege with the ANY and WITH ADMIN OPTION clauses, this statement removes only the ability to administer the system privilege from `Frank`. He can continue to change the password of any user in the database.

```
REVOKE ADMIN OPTION FOR CHANGE PASSWORD (ANY) FROM Frank
```

This statement removes the ability of `Sally` and `Bob` to change the password of `Jane`, `Joe`, and `Laurel` only:

```
REVOKE CHANGE PASSWORD (Jane, Joe, Laurel) FROM Sally, Bob
```

This statement removes the ability of `Mary` to change the password of any member of the `Sales1` role:

```
REVOKE CHANGE PASSWORD (ANY WITH ROLES Sales1) FROM Mary
```

This statement removes the ability of `Sarah` to change the password of `Joe` or `Sue`, or any member of the `Sales2` role:

```
REVOKE CHANGE PASSWORD (Joe, Sue), (ANY WITH ROLES Sales2) FROM Sarah
```

This statement removes the ability of `Joan` to change the password of any member of the `Marketing1` or `Marketing2` roles:

```
REVOKE CHANGE PASSWORD (ANY WITH ROLES Marketing1, Markeing2) FROM Joan
```

Related Information

REVOKE CHANGE PASSWORD Privilege Statement

5.2.5 Changing a Password – Single Control

A single user can manage the password of another user.

Prerequisites

- The CHANGE PASSWORD system privilege.

Procedure

At a command prompt, type:

```
ALTER USER <userID> IDENTIFIED BY <password>
```

Related Information

ALTER USER Statement
Case-Sensitivity of User IDs and Passwords

5.2.6 Dual Control Password Management Option

The Dual Control Password option requires two administrative users to change the password of a target user, thus ensuring that no single user knows (or controls) the password of the target user.

Two distinct administrative users are required to generate each part of the new password. It is the combination of the two parts that become the new password for the target user. The same user cannot generate both password parts. If the same user attempts to define both password parts, the server displays an error message, and the second password part is not set.

If the server is restarted after the first password part is specified, but before the second password part is specified, the first password part is not lost. When the second password part is specified by a different user, the dual password change process completes successfully. The target user can then log in using the combined password parts.

Once initiated, generation of the dual passwords for the target user can be canceled by specifying "NULL" as the password, as long as the user has been granted the CHANGE PASSWORD system privilege, and the right to manage the password of the target user.

Each administrative user setting a password part must notify the target user of the new password part, and indicate whether it is the first or second part. To use the password, the target user enters the dual password in first part, second part order. There is a 127-character limit for each part.

If the target user is not logged in when the dual password change process completes, he or she simply logs in. Once the dual password is accepted, the user is immediately prompted to change his or her password. This provides the final level of password security. If the user is already logged in when the dual password change process completes, the user can use the `ALTER USER` or `GRANT CONNECT` statements, or the `sp_password` or `sp_iqpassword` system procedures to change the password. At the prompt for the current password, enter the new dual part passwords, not the password originally entered for the current session.

The Change Password Dual Control option is enabled in a login policy.

In this section:

Related Information

ALTER USER Statement
GRANT CONNECT Privilege Statement
(Deprecated) sp_iqpassword Procedure
Case-Sensitivity of User IDs and Passwords

5.2.6.1 Enabling Dual Control for Changing Passwords

Require input from two administration users to change the password of another user.

Prerequisites

- Requites the `MANAGE ANY LOGIN POLICY OPTION` system privilege.

Context

Dual control for managing passwords is a configurable option in a login policy. By default, it is disabled (OFF).

Procedure

To enable the option, execute:

```
ALTER LOGIN POLICY <policy-name> CHANGE_PASSWORD_DUAL_CONTROL=ON
```

Related Information

ALTER LOGIN POLICY Statement

5.2.6.2 Changing a Password – Dual Control

Two users are required to manage the password of another user.

Prerequisites

- The CHANGE PASSWORD system privilege.
- The managing user has been granted the right to change the password of the target user.
- The CHANGE_PASSWORD_DUAL_CONTROL option is enabled in the login policy of the managing user.

Procedure

1. At a command prompt, the first managing user enters:

   ```
   ALTER USER <userID> IDENTIFIED FIRST BY <password_part1>
   ```

2. At a command prompt, the second managing user enters:

   ```
   ALTER USER <userID> IDENTIFIED LAST BY <password_part1>
   ```

Example

Assuming login policy `Sales1` has the `CHANGE_PASSWORD_DUAL_CONTROL` option enabled, `User3` is assigned `Sales1`, and `User1` and `User2` have been granted the necessary privileges to change the password of `User3`, these statements set the two password parts for `User3` to `<NewPassPart1>` and `<NewPassPart2>`:

User1 types:

```
ALTER USER user3 IDENTIFIED FIRST BY NewPassPart1
```

User2 types:

```
ALTER USER user3 IDENTIFIED LAST BY NewPassPart2
```

Related Information

ALTER USER Statement
Case-Sensitivity of User IDs and Passwords
Resetting the DBA Password (Command Line) [page 19]

6 Data Confidentiality

You can secure communications between a client and the SAP IQ server, or between an SAP IQ client and the database server using Transport Layer Security (TLS).

SAP IQ allows you to encrypt your database or columns.

Support of Kerberos authentication, and column encryption is included in the separately licensed SAP IQ Advanced Security Option.

In this section:

Related Information

Column Encryption in SAP IQ [page 234]

6.1 Database Encryption and Decryption

Make it more difficult for someone to decipher the data in your database.

You can choose to secure your database either with simple obfuscation or with strong encryption.

The database administrator has control over four aspects of strong encryption, including:

- What is encrypted (database, dbspaces, individual tables, transaction log, transaction log mirror)
- The encryption key
- The protection of the encryption key
- Encryption algorithm

If your database is obfuscated or encrypted, compressing it with a tool such as WinZip does not result in a file that is significantly smaller than the original database file.

In this section:

6.1.1 Simple Obfuscation Versus Strong Encryption

When specifying database encryption algorithm, SIMPLE (obfuscation) or a specific strong encryption algorithm is chosen.

Simple Obfuscation

SIMPLE should not be considered encryption. It is obfuscation and is not cryptographically secure. The SIMPLE algorithm is intended to make it difficult, but not impossible, for someone using a disk utility to casually inspect the contents of your database. Simple obfuscation does not require a key (password) to encode the database.

Strong Encryption (AES and ARIA)

Strong encryption makes a database unusable without a key (password). The data in the database is secure from inspection. An algorithm encrypts the information contained in your database and transaction log files so it cannot be read.

The algorithms used to implement strong encryption are AES and ARIA.

You can indicate the use of a 128-bit encryption algorithm using the AES keyword. You can indicate the use of a 256-bit encryption algorithm using the AES256, AES256CTR, ARIA256, or ARIA256CTR keywords. This encryption technology is included and does not require a separate license.

You can also indicate the use of a FIPS-certified AES module for strong encryption by specifying one of the AES_FIPS (128-bit algorithm) or AES256_FIPS (256-bit algorithm) keywords. When the database server is started with the -fips option, you can start databases encrypted with any of the AES, AES256, AES_FIPS, AES256_FIPS, or AES256CTR_FIPS strong encryption algorithms, but not databases encrypted with the SIMPLE algorithm. Unencrypted databases can also be started on the server when -fips is specified.

All strong encryption technologies are subject to export regulations.

Related Information

Encryption Algorithm Aliases [page 158]

6.1.2 How to Encrypt a Database

There are several ways to encrypt a database.

To create an encrypted database

You can use the following:

- The Initialization utility (iqinit), with options such as -ep, -ek, or -ea to enable strong encryption.
- CREATE DATABASE statement.

You cannot encrypt an existing database.

Related Information

iqinit Database Administration Utility
CREATE DATABASE Statement

6.1.3 Encryption Algorithm Aliases

The encryption algorithms for SAP IQ database encryption have multiple aliases for compatibility across systems.

Encryption Algorithm	Supported Aliases	SAP HANA Database-Compatible Alias	Description
AES	AES128; AES_128; AES-128; AES-128-CBC	AES-128-CBC	AES 128-bit algorithm with CBC block cipher mode
AES_FIPS	AES128_FIPS; AES-128_FIPS; AES-128_FIPS; AES-128-CBC_FIPS		FIPS-certified version of the AES 128-bit algorithm with CBC block cipher mode
AES256	AES-256-CBC; AES-256; AES_256; AES_256_CBC	AES-256-CBC	AES 256-bit algorithm with CBC block cipher mode
AES256_FIPS	AES-256-CBC_FIPS; AES-256_FIPS; AES_256_FIPS; AES_256_CBC_FIPS		FIPS-certified version of the AES 256-bit algorithm with CBC block cipher mode
AES256CTR	AES-256-CTR; AES_256_CTR; AES256_CTR; AES256-CTR	AES-256-CTR	AES 256-bit algorithm with CTR block cipher mode
AES256CTR_FIPS	AES-256-CTR_FIPS; AES_256_CTR_FIPS; AES256_CTR_FIPS; AES256-CTR_FIPS		FIPS-certified version of the AES 256-bit algorithm with CTR block cipher mode
ARIA256	ARIA_256_CBC; ARIA-256; ARIA_256; ARIA256CBC; ARIA256_CBC	ARIA-256-CBC	ARIA 256-bit algorithm with CBC block cipher mode
ARIA256CTR	ARIA-256-CTR; ARIA_256_CTR; ARIA256_CTR; ARIA256-CTR	ARIA-256-CTR	ARIA 256-bit algorithm with CTR block cipher mode

6.1.4 Creating an Encrypted Database (SQL)

Encrypt a database during creation.

Prerequisites

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gu database server option.

Context

> ⚠ Caution
>
> For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

Procedure

1. In Interactive SQL, connect to an existing database.
2. Execute a CREATE DATABASE statement that includes the ENCRYPTED clause and the KEY and ALGORITHM options.

Results

An encrypted database is created.

Example

For example, the following statement creates a database file named `myencrypteddb.db` in the `c:\temp\` directory using FIPS-certified 128-bit AES encryption and the specified encryption key. The DBA user ID and password are specified and a transaction log is enabled.

```
CREATE DATABASE 'c:\\temp\\myencrypteddb.db'
DBA USER 'DBA'
DBA PASSWORD 'passwd'
TRANSACTION LOG ON
ENCRYPTED ON
  KEY '0kZ2o52AK#'
  ALGORITHM 'AES_FIPS';
```

Related Information

CREATE DATABASE Statement

6.1.5 Creating an Encrypted Database (iqinit Utility)

Encrypt the database when you create the database.

Context

> ⚠ Caution
>
> For strongly encrypted databases, store a copy of the key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

Procedure

Run the iqinit utility to create a database.

- To encode the database with strong encryption, include -ek or -ep options to specify the encryption key.

Results

An encoded database is created.

Example

- The following command creates a strongly-encrypted database and specifies the encryption key and algorithm.

  ```
  iqinit -dba DBA,passwd -ek "0kZ2o56AK#" -ea AES_FIPS myencrypteddb.db
  ```

  To start this database, run the following command:

  ```
  start_iq myencrypteddb.db -ek "0kZ2o56AK#"
  ```

Next Steps

When starting a strongly-encrypted database, you must specify the encryption key.

Related Information

iqinit Database Administration Utility

6.1.6 Security: Keys For Encrypted Databases

Learn how to increase the security of the encryption keys used with strongly encrypted databases and tables.

When you encrypt a database or a database table using strong encryption, you must specify an encryption key (password).

Specifying an Encryption Key

The encryption key must be supplied with the -ek option each time the database is started. For example, clients are required to specify the encryption key each time they start the database. If you are concerned about providing your clients with the encryption key, then always have the database administrator start the database.

You can choose whether the encryption key is entered at a command prompt (the default) or into a prompt box. Choosing to enter the key in a prompt box provides an extra measure of security because the key is never visible in plain sight. When starting the database, specify the -ep database option, to be prompted for the encryption key.

Choosing an Encryption Key

When choosing an encryption key, it is best to choose a value that cannot be easily guessed. As well, including a combination of numbers, letters, and special characters decreases the chances of someone guessing the key. Encryption keys are always case sensitive, and they cannot contain leading or trailing spaces or semicolons. Encryption keys can be of arbitrary length. Longer keys are better because a shorter key is easier to guess

Storing Your Encryption Keys

Lost or forgotten keys result in completely inaccessible databases.

> ⚠ Caution
>
> Store a copy of your encryption key in a safe location. If you lose the encryption key, there is no way to access the data, even with the assistance of Technical Support. The database must be discarded and you must create a new database.

6.1.7 Performance Issues When Using Encryption

Performance is slower when the database is encrypted. The performance impact depends on how often pages are read from or written to disk, and can be minimized by ensuring that the server is using an adequate cache size.

You can increase the starting size of the cache with the -c option when you start the server. For operating systems that support dynamic resizing of the cache, the cache size that is used may be restricted by the amount of memory that is available; to increase the cache size, increase the available memory.

6.2 IPv6 Support

SAP IQ supports Internet Protocol version 6 (IPv6), which contains addressing and control information to route packets over the Internet.

IPv6 supports two^{128} unique IP addresses, which is a substantial increase over the number of addresses supported by its predecessor IPv4. SAP IQ supports both IPv4 and IPv6 addresses anywhere you can specify an IP address on the client or server.

JDBC and ODBC classes support the use of IPv6 addresses for remote data access.

6.3 Digital Certificates

You need digital certificates to set up transport-layer security.

Obtain certificates from a certificate authority, or create them using the Certificate Creation utility (createcert).

Certificates From the Operating System Certificate Store

By default, secure connections use your computer's operating system certificate store to obtain a trusted certificate for secure connections.

For all secure connections except HTTPS on Windows operating systems, stored certificates are cached, with the cache reloading every 24 hours. If a required certificate is installed within 24 hours after the first secure connection, then the connection requiring that certificate fails until the cache is reloaded. To make the certificate accessible before 24 hours, restart the server.

On Windows, access the certificate store by running the following command at a command prompt:

```
certmgr.msc
```

Certificate Creation Utility

Use the Certificate Creation utility (createcert), to generate X.509 certificate files using RSA.

Certificate Viewer Utility

Use the Certificate Viewer utility (viewcert), to read X.509 certificates using RSA.

Certificates For Server Authentication

Obtain a certificate from a certificate authority or use the Certificate Creation utility (createcert) to create certificate files for server authentication. In each case, create an identity file and a certificate file.

For server authentication, you create a server identity file and a certificate file to distribute to clients.

Certificate Configurations

The certificate can be self-signed or signed by a commercial Certificate Authority.

Self-signed certificates

Self-signed server certificates can be used for simple setups.

Root certificates

A root certificate can be used to sign server certificates to improve data integrity and extensibility for multi-server deployments.

- You can store the private key used to sign server certificates in a secure central location.
- For server authentication, you can add database servers without reconfiguring clients.

Commercial Certificate Authorities

You can use a third-party Certificate Authority instead of a root certificate. Commercial Certificate Authorities have dedicated facilities to store private keys and create high-quality server certificates.

In this section:

Related Information

Certificate Creation Utility (createcert)
Certificate Viewer Utility (viewcert)

6.3.1 Self-signed Root Certificates

Self-signed root certificates can be used for simple setups involving a single database server.

> → Tip
>
> Use certificate chains or commercial certificate authorities if you require multiple server identity files. Certificate authorities provide extensibility and a higher level of certificate integrity with dedicated facilities to store root private keys.

Certificate

For server authentication certificates, the self-signed certificate is distributed to clients. It is an electronic document including identity information, the public key of the server, and a self-signed digital signature.

Identity file

For server authentication certificates, the identity file is stored securely with a database server. It is a combination of the self-signed certificate (that is distributed to clients) and the corresponding private key. The private key gives the database server the ability to decrypt messages sent by the client in the initial handshake.

6.3.2 Certificate Chains

If you require multiple identity files, you can improve security and extensibility by using certificate chains instead of self-signed certificates.

Certificate chains require a root certificate to sign identities. You can create this root certificate yourself or obtain one from a certificate authority.

Benefits of Using Certificate Chains

Certificate chains provide the following advantages:

Extensibility

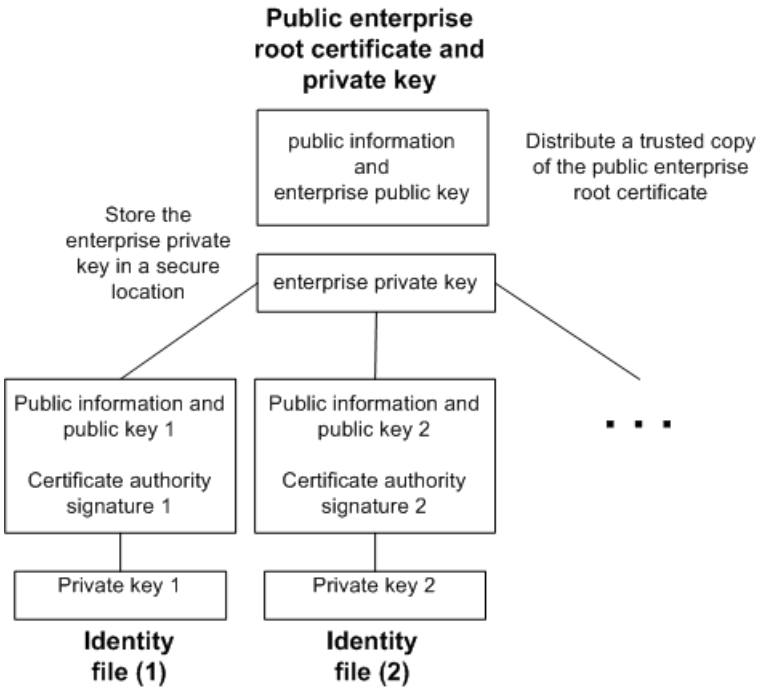
For server authentication, you can configure clients to trust any certificate signed by a specific root certificate.

. If you add a new database server, clients do not require a copy of the new certificate.

Security

The root certificate's private key is not in the identity file. Storing the root certificate's private key in a high-security location, or using a Certificate Authority with dedicated facilities, protects the integrity of server authentication.

The following diagram provides the basic root certificate architecture.



Using Certificates in a Multi-Server Environment

To create certificates used in a multi-server environment:

- Generate a public root certificate and private key.
  Store the root private key in a secure location, preferably a dedicated facility.
  For server authentication, you distribute the public root certificate to clients.
- Use the root certificate to sign identities.
  Use the public root certificate and root private key to sign each identity. For server authentication, the identity file is used for the server.

You can also use a third-party Certificate Authority to sign your server certificates. Commercial Certificate Authorities have dedicated facilities to store private keys and create high-quality server certificates.

In this section:

Root Certificates [page 166]
Root certificates improve data integrity and extensibility for multi-server deployments.

Signed Identity Files [page 166]
You can use a root certificate to sign database server identity files.

6.3.2.1 Root Certificates

Root certificates improve data integrity and extensibility for multi-server deployments.

- You can store the private key used to create trusted certificates in a dedicated facility.
- For database server authentication, you can add database servers without reconfiguring clients.

To set up root certificates, you create the root certificate and the private key that you use to sign identities.

6.3.2.2 Signed Identity Files

You can use a root certificate to sign database server identity files.

For database server authentication, you generate identity files for each server. Since these certificates are signed by a root certificate, you use the createcert -s option.

6.3.3 Globally Signed Certificates

Globally signed certificates are high-quality certificates that are used to sign your certificate requests.

These high-quality certificates are created and managed by a commercial Certificate Authority.

Globally signed certificates have the following advantages:

- For inter-company communication, common trust in an outside, recognized authority may increase confidence in the security of the system. A Certificate Authority must guarantee the accuracy of the identification information in any certificate that it signs.
- Certificate Authorities provide controlled environments and advanced methods to generate certificates.
- The private key for the root certificate must remain private. Your organization may not have a suitable place to store this crucial information, whereas a Certificate Authority can afford to design and maintain dedicated facilities.

Setting up Globally Signed Certificates

To set up globally signed identity files, you:

- Create a certificate request using the createcert utility with the -r option.
- Use a Certificate Authority to sign each request. You can combine the signed request with the corresponding private key to create the server identity file.

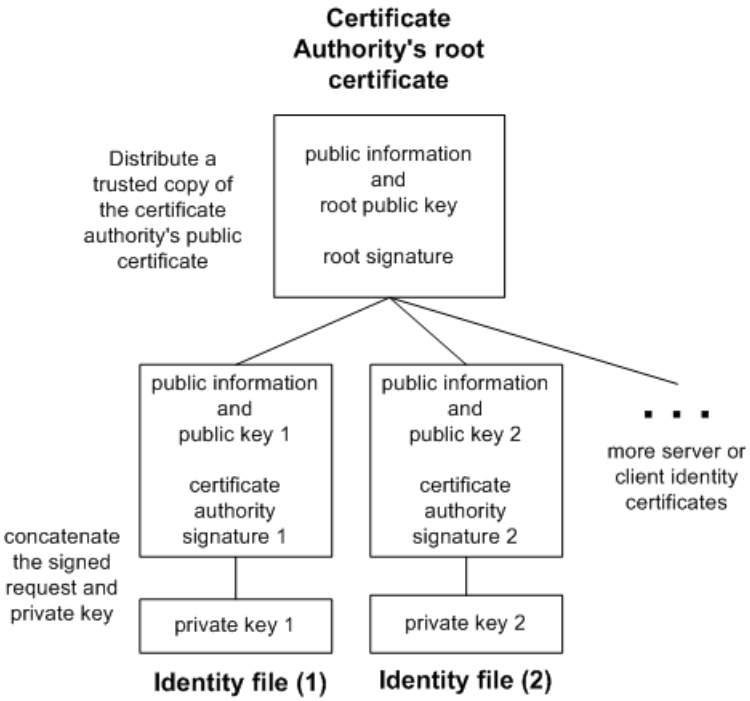
> **Note**
>
> You might be able to globally sign a root certificate. This is only applicable if your Certificate Authority generates certificates that can be used to sign other certificates.

In this section:

6.3.3.1 Globally Signed Identity Files

You can use globally signed certificates directly as server identity files.

The following diagram shows the configuration for multiple identity files:



Reference the server identity file and the password for the private key on the start_iq command line.

6.3.3.2 Client Trust Setup for the Certificate Authority's Certificate

For server authentication, you must ensure that clients contacting your server trust the root certificate in the chain.

For globally signed certificates, the root certificate is the Certificate Authority's certificate.

When using a globally signed certificate, each client must verify certificate field values to avoid trusting certificates that the same Certificate Authority has signed for other clients.

6.4 How to Set Up Transport Layer Security

Set up transport layer security on your system.

Prerequisites

Obtain digital certificates. You need identity files and certificate files. The server identity file contains the server's private key and should be stored securely with the database. You distribute the server certificate file to your clients.

You can buy certificates from a certificate authority or you can use the Certificate creation utility (createcert). Functionality to create certificates, which is especially useful for development and testing, is provided.

Procedure

1. If you are setting up transport layer security for SAP IQ client/server applications:

   Start the SAP IQ database server with transport layer security

   Use the -ec database server option to specify the type of security, the server identity file name, and the password to protect the server's private key.

   If you also want to allow unencrypted connections over shared memory, specify the -es option.

   TDS connections support the TLS protocol. To prevent unencrypted connections from using the TDS protocol, specify the tcpip option `-x tcpip(TDS=ENCRYPTED)`.

   Configure client applications to use transport layer security

   Specify the path and file name of trusted certificates using the Encryption connection parameter [ENC].
2. If you are setting up transport layer security for SAP IQ web services:

   Start the SAP IQ database server with transport layer security

   Use the -xs database server option to specify the type of security, the server identity file name, and the password to protect the server's private key.

   TDS connections support the TLS protocol. To prevent unencrypted connections from using the TDS protocol, specify the tcpip option `-x tcpip(TDS=ENCRYPTED)`.

   Configure browsers or other web clients to trust certificates

   Encrypt SAP IQ web services.
3. If you are setting up an SAP IQ multiplex database server:

- INC and MIPC connections determine which TLS connection parameters to use from the contents of the -ec server option.
- Set the TRUSTED_CERTIFICATES_FILE option to the appropriate Certificate Authority.

Results

You have set up transport layer security on your system.

In this section:

Related Information

-ec Database Server Option
-es Database Server Option
-x Database Server Option
-xs Database Server Option
Encryption (ENC) Connection Parameter

6.4.1 TLS Support

Transport Layer Security (TLS) encryption versions *1.0*, *1.1*, *1.2*, and *1.3* are supported.

TLS 1.2 and 1.3 are selected by default with the server and client negotiating the highest protocol they both support. The min_tls_version network protocol option can be used to change allowed protocols.

Cipher Suites

The following cipher suites (in server preference order) are supported.

- When the minimum TLS version is *1.3*, the supported cipher suites are:
  - TLS_AES_128_GCM_SHA256 (with ECDHE_RSA key exchange)
  - TLS_AES_256_GCM_SHA384 (with ECDHE_RSA key exchange)
- When the minimum TLS version is *1.2*, the supported cipher suites are:
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp384r1)
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp384r1)
  - TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048)
  - TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048)
- When the minimum TLS version is *1.1* or *1.0*, the supported cipher suites are:
  - TLS *1.1* and *1.0* cipher suites:
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp384r1)
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp384r1)
    - TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048)
    - TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048)
  - TLS *1.2* cipher suites:
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (secp384r1)
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (secp384r1)
    - TLS_RSA_WITH_AES_128_GCM_SHA256 (rsa 2048)
    - TLS_RSA_WITH_AES_256_GCM_SHA384 (rsa 2048)
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (secp384r1)
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (secp384r1)
    - TLS_RSA_WITH_AES_128_CBC_SHA (rsa 2048)
    - TLS_RSA_WITH_AES_256_CBC_SHA (rsa 2048)

RSA Encryption

Noncertified RSA encryption is included with the software and can be used for client/server communication, synchronization, and Web services. This version isn't FIPS-certified.

FIPS-Certified RSA Encryption

FIPS-certified (FIPS PUB 140-2) RSA encryption is available.

Related Information

Encryption (ENC) Connection Parameter
min_tls_version Protocol Option

6.4.2 TLS Cipher Suite Aliases

Alias names and descriptions for supported TLS cipher suites.

Use these aliases with the set_ciphers network protocol option to specify a broad class of ciphers instead of specifying each one individually.

Cipher Suite Alias	Description
ALL	All supported cipher suites in a preferred order, cipher suites with highest security first
PFS	Perfect forward secrecy: key agreement with ephemeral keys
HIGH	High security cipher suites (except PFS)
eAES	AES encryption cipher sites
eAES128	AES 128 encryption cipher sites
eAES256	AES 256 encryption cipher sites
eAES_CBC	AES in CBC mode encryption cipher sites
eAES128_CBC	AES 128 in CBC mode encryption cipher sites
eAES256_CBC	AES 256 in CBC mode encryption cipher sites
eAES_GCM	AES in GCM mode encryption cipher sites
eAES128_GCM	AES 128 in GCM mode encryption cipher sites
eAES256_GCM	AES 256 in GCM mode encryption cipher sites
eARIA	ARIA encryption cipher sites
eARIA128	ARIA 128 encryption cipher sites
eARIA256	ARIA 256 encryption cipher sites
eARIA_GCM	ARIA in GCM mode encryption cipher sites
eARIA128_GCM	ARIA 128 in GCM mode encryption cipher sites
eARIA256_GCM	ARIA 256 in GCM mode encryption cipher sites
eSEED	SEED encryption cipher suites
mSHA1	Cipher sites with SHA-1 message authentication

Key Exchange Options

Options to select key exchange algorithms, including elliptic curve (EC) for Elliptic-curve Diffie-Hellman (ECDH) and Elliptic-curve Diffie-Hellman Ephemeral (ECDHE) key exchange.

Option	Description
EC_HIGH	Enables high strength elliptical curve (ECDHE) key exchanges.
+EC_OPT	Turns on elliptical curve optimizations (off by default).
EC_X25519	Curve25519, offering 128 bits of security
EC_P256	Also known as secp256r1, offering 128 bits of security (256-bit key size)
EC_X448	Curve448, offering 224 bits of security
EC_P384	Also known as secp384r1, offering approximately 192 bits of security (384-bit key size)
EC_P521	Also known as secp521r1, offering approximately 256 bits of security (256-bit key size)

6.4.3 Set TLS Cipher Suites

Use the min_tls_version and set_ciphers network protocol options to control the TLS cipher suites used by your connection.

Control Cipher Suites Using min_tls_version

Setting the min_tls_version network protocol option to either *10*, *11*, *12*, or *13* affects which TLS cipher suites the server offers the client. See TLS Support [page 169] for details.

Control Cipher Suites Using set_ciphers

For finer control over the cipher suites offered by the server – including the ability to block broken ciphers – use the set_ciphers network protocol option. For details and examples, see set_ciphers Protocol Option, -ec Database Server Option, and -xs Database Server Option.

Related Information

min_tls_version Protocol Option

6.4.4 Check TLS Protocol Version Used by Your Connection

Use the TLSVersion connection property to check what TLS protocol version your connection is secured with.

For example, run:

```
SELECT CONNECTION_PROPERTY ( 'TLSVersion' );
```

See TLSVersion in List of Connection Properties for more details.

Related Information

min_tls_version Protocol Option

6.4.5 Check TLS Ciphers Used by Your Connection

Use the TLSCipher network connection property to return the cipher used for the connection. Use the TLSSuites network connection property to list the cipher suites the server is permitted to use.

For example, run these two statements:

```
SELECT CONNECTION_PROPERTY ( 'TLSCipher' );
```

```
SELECT CONNECTION_PROPERTY ( 'TLSSuites' );
```

See TLSCipher and TLSSuites in List of Connection Properties for more details.

Related Information

min_tls_version Protocol Option

6.5 Using Client-side Certificates for TLS

Use client-side certificates so that clients connect to your database server only if they use a certificate signed by a trusted source.

Context

In addition to encrypting the communication stream and requiring server-side authentication using certificates, you can also use a second set of client-side certificates so that clients can only connect to your server if they use a certificate signed by a trusted source. This is often referred to as mutual authentication or two-way authentication.

This example creates the necessary certificates for mutual authentication and assumes you are using a single root certificate to sign the client certificate and server certificate. In a production environment you may wish to use two different root certificates and purchase database server and client certificates from a certificate authority.

Procedure

1. Use the Certificate Creation utility (createcert) to create the self-signed root certificate. For example, run the following command all on one line:

   ```
   createcert -b 2048 -x -ca 1 -co myroot.crt -io myroot.id -ko myroot.pk -kp
   root_pw -m 123 -sc CA -sst Ontario
    -sl Waterloo -so MyCompany -sou MyUnit -scn "Sample root certificate" -u 6,7
   -v 10
   ```

2. Use createcert to create the database server certificate. For example, run the following command all on one line, replacing MyServerHost with the host name of the computer that the database server runs on:

   ```
   createcert -b 2048 -c myroot.crt -ck myroot.pk -cp root_pw -ca 0 -co
   myserver.crt -io myserver.id -ko myserver.pk -kp server_pw -m 124
   -sc CA -sst Ontario -sl Waterloo -so MyCompany -sou MyUnit -scn MyServerHost
   -u 1,3,4,5 -v 10
   ```

3. Use createcert to create the client certificate. For example, run the following command all on one line:

   ```
   createcert -b 2048 -c myroot.crt -ck myroot.pk -cp root_pw -ca 0 -co
   myclient.crt -io myclient.id -ko myclient.pk -kp client_pw -m 125
    -sc CA -sst Ontario -sl Waterloo -so SAP -sou MyUnit -scn MyClientHost -u
   1,3,4,5 -v 10
   ```

   MyClientHost is arbitrary and does not need to match any host name.

4. Start the database server and include the identity and identity_password options to specify the database server's identity file and the trusted_certificate option to specify the root certificate. For example, run the following command all on one line:

   ```
   start_iq -n MyDBServer MyDatabase.db -x tcpip -ec
   "TLS(identity=myserver.id;identity_password=server_pw;trusted_certificate=myro
   ot.crt)"
   ```

   The root certificate is the certificate that signs the client's certificate.
5. Test the client application connection. For example, using the Ping utility (dbping), run the following command all on one line.

   ```
   dbping -d -c
   "UID=DBA;PWD=sql;HOST=MyServerHost;DBN=MyDatabase;ENC=TLS(trusted_certificate=
   myroot.crt;identity=myclient.id;identity_password=client_pw;certificate_name=M
   yServerhost)"
   ```

   The identity and identity_password options specify the client's identity file and the trusted_certificate option specifies the root certificate for the database server (the certificate that signs the client's certificate).

Results

You have created a trusted certificate for your client and your database server has accepted your client application connection by verifying the client certificate.

7 Utility Database Server Security

SAP IQ includes a phantom database, called the utility database that has no physical representation, and which can contain no data.

The utility database can run on any SAP IQ server. In SAP IQ Cockpit, the server for the utility database is known as the Utility Server.

The utility database permits a narrow range of specialized functions. It enables you to execute database file manipulation statements such as `CREATE DATABASE` and `DROP DATABASE` without first connecting to a physical database.

You can also retrieve database and connection properties from the utility database. These properties apply to databases you create when connected to the utility database.

One of your configuration tasks is to set up security for the utility database and its server. You must decide:

- Who can connect to the utility database, and
- Who can execute file administration statements.

In this section:

7.1 Defining the Utility Database Name When Connecting

You cannot specify a database file when starting the utility database, because no database file is associated with that database. You must specify the database name when connecting.

Procedure

Specify `utility_db` as the database name when connecting to the utility database. Passwords must be 6 bytes in length.

For example:

```
dbisqlc -c "uid=dba;pwd=<passwd>;eng=myserver;dbn=utility_db"
```

Results

> **Note**
>
> When you connect to the utility database to create an IQ database that uses Windows raw partitions, there is a syntax difference in the IQ PATH. For example, to specify a Windows raw partition on device I: for the utility database, you can use the specification "\\.\I:" On other IQ databases, you must double the slash characters, so that the same device is specified as "\\\\.\\I:". The backslash character is treated as an escape character in IQ databases but as a normal character in the utility database.

7.2 Defining the Utility Database Password

Define the user ID and password for the utility database (utility_db).

The utility database has a default user ID of DBA, but no default password. The password is defined during start up, using the -su database server option to set the password. If no user ID is specified, the default user ID is used.

```
start_iq -su <user-ID>,<password> -n utility_db
```

Example

The following command starts the utility database with the default user `DBA` (since not specified) and password `MyPassword1`:

```
start_iq -su MyPassword1 -n utility_db
```

The following command starts the utility database with user `User1` and password `MyPassword1`:

```
start_iq -su User1,MyPassword1 -n utility_db
```

7.3 Permission to Execute File Administration Statements

A separate level of security, which controls the creating and dropping of databases, provides additional database security. The `-gu` database server command line option controls who can execute the file administration statements.

There are four levels of permission for the use of file administration statements: `all`, `none`, `DBA`, and `utility_db`. The `utility_db` level permits a user who can connect to the utility database to use the file administration statements.

-gu Switch Value	Effect	Applies To
all	Anyone can execute file administration statements	Any database including the utility database
none	No one can execute file administration statements	Any database including the utility database
DBA	Only users with the SERVER OPERATOR system privilege can execute file administration statements	Any database including the utility database
utility_db	Only the users who can connect to the utility database can execute file administration statements	Only the utility database

Examples

(Sun, HP, Linux, and Windows platforms) To permit only the user knowing the utility database password to connect to the utility database and create or delete databases, start the server at the command line:

```
start_iq -n testsrv -gu utility_db
```

(AIX) To permit only the user knowing the utility database password to connect to the utility database and create or delete databases, start the server at the command line:

```
start_iq -n testsrv -gu utility_db -iqmt 256
```

Assuming that the utility database password was set to `IQ&Mine49` during installation, the following command starts the Interactive SQL utility as a client application, connects to the server named `testsrv`, loads the utility database, and connects the user:

```
dbisql -c "uid=DBA;pwd=IQ&Mine49;dbn=utility_db;eng=testsrv"
```

Executing this statement successfully connects you to the utility database, and you can now create and delete databases.

> **Note**
>
> The database name, user ID, and password are case-sensitive. Make sure that you specify the same case in the `dbisql` command and the `util_db.ini` file.

8 Data Security

Since databases may contain proprietary, confidential, or private information, it is important that you ensure that the database and the data in it are designed for security.

In this section:

8.1 General Security Tips

There are many actions you can take to improve the security of your data.

Choose Passwords Carefully

Never deploy databases with short or easy-to-guess passwords. For more information, see Passwords [page 144].

Restrict Use of Super-Users

Avoid creating and granting roles that possess all privileges, roles, and administration rights (that is, avoid super-users). Instead, create roles with logical groupings of privileges and rights, and then grant those roles judiciously. If you do create super-users, consider using them only when absolutely necessary, and store their passwords in a secure location, such as a safe, so that you can retrieve them when needed.

Consider giving your database administrators two user IDs: one with all privileges and one with limited privileges, so they can connect to the one with all privileges only when necessary.

For more information, see DBA (Default) User [page 12].

Be Cautious of Granting:

- Any privileges with the word "ANY" in their names such as ALTER ANY OBJECT. These give broad access across a database.
- Database, DBspaces, Mirror Server, Replication, and Server Operators privileges that relate to administration of running databases.
- External environments privileges which allow remote code execution.
- Files, directories and disk drives privileges that access server local file system (even when disk sandboxing). READ/WRITE CLIENT FILE are a risk when allowed in SECURITY DEFINER procedures.
- Auditing, LDAP, Roles, and Users and login management privileges which determine authorization and authentication.

- Email privileges due to ability to generate spam.
- ACCESS USER PASSWORD exposes SYSUSERPASSWORD password hashes and SYSEXTERNALLOGINPASSWORD encrypted remote_password. This privilege should only be granted when there is a need to unload a complete database.

Use Secured Database Features

Feature keys can be created to provide custom set of features to specific users. The list of features can be found in sa_server_option System Procedure. For further information, see Creating Secured Feature Keys [page 181].

The -sf database server option lets you enable and disable features for all databases running on a database server. The features you can disable include the use of external stored procedures, Java, remote data access, and the ability to change the request log settings.

Disabling Database Features

The -sf database server option specifies a list of features that are disabled for databases running on the database server so they are not available to client applications or stored procedures, triggers, or events defined within the databases. This can be useful when you are starting a database that is not your own that may attempt unwanted operating system interactions such as file transfers between the database client and server.

8.2 System Secure Features

You can make system secure features inaccessible to databases running on a database server.

When a feature is secured (made inaccessible), it's unavailable for use by client applications, database-defined stored procedures, triggers, and events. Secure feature settings apply to all databases that are running on the selected database server. Secure features are useful when you need to start a database that might contain embedded logic that you're unsure about, such as a virus, or if you want to lock down a database server or database hosted by a third-party vendor. The `-sf` database server option allows you to specify which features you want to secure for databases running on the database server.

Secure Feature Keys

A `system secure feature key` is created by specifying the `-sf` database server option when creating the database server. Use the `sa_server_option` system procedure to alter whether features are secured or unsecured once the database server is running.

Once you've created a system secure feature key, you can create `customized secure feature keys` that are assigned to a specific user, limiting users' access to only the features secured by the administrator for that key.

Customized secure feature keys are managed by select system procedures.

In this section:

Control the database features available to users, by using the secure features database server option (-sf) to specify the features that users are prevented from accessing on the database server.

Related Information

sa_server_option System Procedure

8.2.1 Creating Secured Feature Keys

Control the database features available to users, by using the secure features database server option (-sf) to specify the features that users are prevented from accessing on the database server.

Prerequisites

You must have the SERVER OPERATOR system privilege and have access to the MANAGE_KEYS feature.

Context

Secured feature settings apply to all databases running on a database server.

The secure features option (-sf) controls the availability of such features as:

- Server-side backups
- External stored procedures
- Remote data access
- Web services

The -sk option specifies a SYSTEM secured feature key that manages access to secured features for a database server. To alter the list of secured features once the database server is running, use the sa_server_option system procedure. To alter a customized secured feature key once the database server is running, use the sp_alter_secure_feature_key system procedure.

The sp_create_secure_feature_key system procedure creates a customized secured feature key.

The list of features can be found in sa_server_option System Procedure .

Procedure

1. At a command prompt, start the database server using the -sf and -sk options.

For example, the following command starts the database server and secures all features. The command also includes a key that can be used later to allow access to secured features for a connection.

```
start_iq -n secure_server -sf all -sk secretAuthCode mydemo.db
```

2. Connect to the database server:

```
dbisql -c "UID=DBA;PWD=passwd;Host=myhost;Server=secure_server;DBN=mydemo"
```

3. Call the sp_use_secure_feature_key system procedure to specify the SYSTEM secured feature key for the connection. The authorization code to use is specified by the -sk option:

```
CALL sp_use_secure_feature_key ( 'system' , 'secretAuthCode' );
```

4. Change the set of secured features on the server by using the sa_server_option system procedure.

   For example:

```
CALL sa_server_option( 'SecureFeatures', 'all,-remote_data_access' );
```

5. Create a customized secured feature key for a specific user.

   For example, create a customized secured feature key for Bob that allows him to send emails:

```
CALL sp_create_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' ,
'send_email' );
```

   After logging into the database, Bob must run the following command to send emails:

```
CALL sp_use_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' );
```

Results

There is now a SYSTEM secured feature key for the database server, as well as a customized secured feature key that has been assigned to a specific user.

Users of databases running on the database server secure_server are prevented from accessing all secured features except the remote_data_access feature. The user Bob, however, also has access to the send_email feature.

Related Information

sp_alter_secure_feature_key System Procedure
sp_create_secure_feature_key System Procedure
sp_drop_secure_feature_key System Procedure
sp_list_secure_feature_keys System Procedure
sp_use_secure_feature_key System Procedure

8.3 Security with Views and Procedures

You can use views and stored procedures to tailor privileges to suit the needs of your enterprise.

For databases that require a high level of security, there are limitations on defining privileges directly on tables. Any privilege granted to a user on a table applies to the entire table. You may need to assign privileges more precisely than on a table-by-table basis. For example:

- You do not want to give access to personal or sensitive information stored in an employee table to users who need access to other parts of the table.
- You may want to give sales representatives privileges on a table containing descriptions of sales calls, but only allow them to update privileges to their own calls.

In this section:

8.3.1 Views Provide Tailored Security

Use views to give users access to only one portion of a table.

You can define a portion in terms of rows or columns. For example, you may want to disallow a group of users from seeing the Salary column of an Employees table, or you may want to allow a user to see only the rows of a table that he or she has created.

Example 1

The sales manager needs access to information in the database concerning employees in the department. However, there is no reason for the manager to have access to information about employees in other departments.

Create a user ID for the sales manager, create views that provide the information needed, and grant the appropriate privileges to the sales manager user ID.

1. As a user with the `MANAGE ANY USER` system privilege, create the new user ID using the `GRANT` statement. Enclose the user name in quotation marks, because it is a SQL keyword.

```
CONNECT "<user_name>"
IDENTIFIED by <password>;
GRANT CONNECT
TO SalesManager
IDENTIFIED BY sales
```

2. Define a view that looks only at sales employees. Identify the table as "HR_USER". Employees, with the owner of the table explicitly identified, so that the SalesManager user ID can use the view. Otherwise, when SalesManager uses the view, the SELECT statement refers to a table that the user ID does not recognize.

```
CREATE VIEW emp_sales AS
SELECT EmployeeID, GivenName, Surname
FROM "HR_USER".Employees
WHERE DepartmentID = 200
```

3. Give SalesManager privilege to look at the view. Use the same command to grant privilege on a view as to grant privilege on a table.

```
GRANT SELECT
ON emp_sales
TO SalesManager
```

Example 2

This example creates a view, which allows the sales manager to look at a summary of sales orders. This view requires information from more than one table for its definition:

1. Create the view.

```
CREATE VIEW order_summary AS
SELECT OrderDate, Region, SalesRepresentative
FROM "GROUPO".SalesOrders
    KEY JOIN "GROUPO".Customers
```

2. Grant privilege for SalesManager to examine this view.

```
GRANT SELECT
ON order_summary
TO SalesManager
```

3. To check that the process has worked properly, connect to the SalesManager user ID and look at the views you have created:

```
CONNECT SalesManager IDENTIFIED BY s@les12345
```

No privileges have been granted to SalesManager to look at the underlying tables. Therefore, these commands produce privilege errors:

```
SELECT * FROM "HR_USER".Employees ;
SELECT * FROM "HR_USER".SalesOrders;
```

These examples show how to use views to tailor SELECT privileges. You can grant INSERT, DELETE, and UPDATE privileges on views in the same way.

In this section:

Guidelines for Using Views [page 185]
There are certain restrictions, both on the SELECT statements you use to create views, and on your ability to insert into, delete from, or update them.

8.3.1.1 Guidelines for Using Views

There are certain restrictions, both on the SELECT statements you use to create views, and on your ability to insert into, delete from, or update them.

Restrictions on SELECT Statements

You cannot use an ORDER BY clause in the SELECT query. A characteristic of relational tables is that there is no significance to the ordering of the rows or columns, and using an ORDER BY clause imposes an order on the rows of the view. You can use the GROUP BY clause, subqueries, and joins in view definitions.

Scalar value subqueries are supported only within the top-level SELECT list (not in a view, derived table, or subquery). Sometimes views or derived tables used in the FROM clause of the top-level SELECT are simple enough that they can be "flattened" up into the top-level SELECT. As a result of this, the preceding rule is actually enforced only for subqueries, nonflattened views, and nonflattened derived tables. For example:

```
CREATE VIEW test_view AS SELECT testkey,(SELECT COUNT(*) FROM tagtests WHERE
tagtests.testkey = testtrd.testkey ) FROM testtrd
```

```
SELECT * FROM test_view
Msg 21, Level 14, State 0:
SQL Anywhere Error -1005004: Subqueries are allowed only as arguments of
comparisons, IN, and EXISTS,
-- (opt_Select.cxx 2101)
```

To develop a view, tune the SELECT query by itself until it provides exactly the results you need in the format you want. Once you have the correct SELECT query, you can add a phrase in front of the query to create the view. For example:

```
CREATE VIEW <viewname> AS
```

Guidelines for Inserting and Deleting from Views

UPDATE, INSERT, and DELETE statements are allowed on some views, but not on others, depending on their associated SELECT statement.

You cannot update, insert into, or delete from views that contain:

- Aggregate functions, such as COUNT(*)
- A GROUP BY clause in the SELECT statement
- A UNION operation

In all these cases, there is no way to translate the UPDATE, INSERT, or DELETE into an action on the underlying tables.

8.3.2 Use Procedures to Provide Tailored Security

Procedures restrict the actions that a user may take.

A user may have `EXECUTE` privilege on a procedure without having any privileges on the table or tables on which the procedure acts.

By default, procedures execute with the privileges of the procedure owner. For a procedure that updates a table, if the procedure owner has `UPDATE` privileges on the table, the user can execute the procedure. The owner of the procedure can restrict the procedure to execute with the privileges of the user executing the procedure by specifying SQL `SECURITY INVOKER` to a `CREATE PROCEDURE` statement.

In this section:

Related Information

CREATE PROCEDURE Statement

8.3.2.1 Setting Up Task-Based Security Restrictions

Disallow all access to the underlying tables, and grant privileges to users or roles to execute certain stored procedures. This approach strictly defines how to control database modifications.

Procedure

1. Create a role for each set of authorized tasks to be performed, and grant the role the applicable system privileges.
2. Grant each of these roles to a single common role.
3. Grant `EXECUTE` privileges on the IQ procedure for performing the authorized tasks to the applicable role.
4. When you create a new user who is to be granted authorized tasks, grant the role created for each authorized task to the user.

8.3.2.2 Granting Users the Privilege to Run Related Stored Procedures

Grant users the system privilege required to run stored procedures. Since most privileges are inherited through role membership, users can inherit the system privilege and the execute privileges for IQ procedures from a role.

Prerequisites

The MANAGE ANY USER or EXECUTE ANY PROCEDURE system privilege.

Procedure

1. Create a role.

   ```
   CREATE ROLE USER_ADMIN_GRP
   ```

2. Grant the MANAGE ANY USER system privilege to the role.

   ```
   GRANT MANAGE ANY USER TO USER_ADMIN_GRP
   ```

3. Grant EXECUTE privilege on SAP IQ stored procedures for user administration to the role.

   ```
   GRANT EXECUTE on sp_iqaddlogin
   to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqcopyloginpolicy
   to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqdroplogin
   to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqmodifyadmin
   to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqmodifylogin
   to USER_ADMIN_GRP
   ```

4. Grant the role to a user. The user inherits the MANAGE ANY USER system privilege and the ability to execute the assigned SAP IQ procedures through membership in the role.

   ```
   GRANT ROLE USER_ADMIN_GRP TO user1
   ```

In this section:

Related Stored Procedures for Role Access [page 188]
You may create roles that grant privileges for various related stored procedures.

8.3.2.2.1 Related Stored Procedures for Role Access

You may create roles that grant privileges for various related stored procedures.

Role Name	System Privilege Granted	Stored Procedure
OPERATOR_GRP	BACKUP DATABASE	sp_iqbackupdetails
	DROP CONNECTION	sp_iqbackupsummary
	CHECKPOINT	sp_iqconnection
	MONITOR	sp_iqsysmon
	ACCESS SERVER LS	
SPACEADMIN_GRP	MANAGE ANY DBSPACE	sp_iqdbspace
	ACCESS SERVER LS	sp_iqdbspaceinfo
		sp_iqdbspaceobjectinfo
		sp_iqemptyfile
		sp_iqestdbspaces
		sp_iqfile
		sp_iqobjectinfo
		sp_iqspaceused

Related Information

sp_iqbackupdetails Procedure
sp_iqbackupsummary Procedure
sp_iqconnection Procedure
sp_iqdbspace Procedure
sp_iqdbspaceinfo Procedure
sp_iqdbspaceobjectinfo Procedure
sp_iqemptyfile Procedure
sp_iqestdbspaces Procedure
sp_iqfile Procedure
sp_iqobjectinfo Procedure
sp_iqspaceused Procedure
sp_iqsysmon Procedure

9 Data Protection and Privacy in SAP IQ

SAP IQ provides the technical enablement and infrastructure to allow you run applications on SAP IQ to conform to the legal requirements of data protection in the different scenarios in which SAP IQ is used.

Introduction to Data Protection

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data privacy regulations, it is necessary to consider compliance with industry-specific legislation in different countries.

SAP provides specific features and functions to support compliance with regard to relevant legal requirements, including data protection. SAP does not give any advice on whether these features and functions are the best method to support company, industry, regional, or country-specific requirements. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments; decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements.

> ⓘ Note
>
> SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and specific data protection-relevant functions, such as simplified blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws will not be covered by a product feature. Definitions and other terms used in this document are not taken from a particular legal source.

Table 1: Glossary

Term	Definition
Blocking	A method of restricting access to data for which the primary business purpose has ended.
Business purpose	A legal, contractual, or in other form justified reason for the processing of personal data. The assumption is that any purpose has an end that is usually already defined when the purpose starts.
Consent	The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.
Deletion	Deletion of **personal data** so that the data is no longer available.
End of business	Date where the business with a data subject ends, for example the order is completed, the subscription is cancelled, or the last bill is settled.
End of purpose (EoP)	End of purpose and start of blocking period. The point in time, when the primary processing purpose ends (e.g. contract is fulfilled).

Term	Definition
End of purpose (EoP) check	A method of identifying the point in time for a data set when the processing of **personal data** is no longer required for the primary **business purpose**. After the **EoP** has been reached, the data is **blocked** and can only be accessed by users with special authorization (for example, tax auditors).
Personal data	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person
Residence period	The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.
Retention period	The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.
Sensitive personal data	A category of personal data that usually includes the following type of information: • Special categories of personal data, such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, biometric data, data concerning health or sex life or sexual orientation, or personal data concerning bank and credit accounts. • Personal data subject to professional secrecy. • Personal data relating to criminal or administrative offenses. • Personal data concerning insurances and bank or credit card accounts.
Where-used check (WUC)	A process designed to ensure data integrity in the case of potential blocking of business partner data. An application's where-used check (WUC) determines if there is any dependent data for a certain business partner in the database. If dependent data exists, this means the data is still required for business activities. Therefore, the blocking of business partners referenced in the data is prevented.

Deleting Personal Data

- Simplified blocking and deletion – When considering compliance with data protection regulations, it is also necessary to consider compliance with industry-specific legislation in different countries. A typical potential scenario in certain countries is that personal data shall be deleted after the specified, explicit, and legitimate purpose for the processing of personal data has ended, but only as long as no other retention periods are defined in legislation, for example, retention periods for financial documents. Legal requirements in certain scenarios or countries also often require blocking of data in cases where the specified, explicit, and legitimate purposes for the processing of this data have ended, however, the data still has to be retained in the database due to other legally mandated retention periods. In some scenarios, personal data also includes referenced data. Therefore, the challenge for deletion and blocking is first to handle referenced data and finally other data, such as business partner data.
- Deletion of personal data – The processing of personal data is subject to applicable laws related to the deletion of this data when the specified, explicit, and legitimate purpose for processing this personal data has expired. If there is no longer a legitimate purpose that requires the retention and use of personal data,

it must be deleted. When deleting data in a data set, all referenced objects related to that data set must be deleted as well. Industry-specific legislation in different countries also needs to be taken into consideration in addition to general data protection laws. After the expiration of the longest retention period, the data must be deleted.

Change Logs

Personal data is subject to frequent changes. Therefore, for revision purposes or as a result of legal regulations, it may be necessary to track the changes made to this data. When these changes are logged, you should be able to check which employee made which change, the date and time, the previous value, and the current value. It is also possible to analyze errors in this way.

SAP IQ Approach to Data Protection

Many data protection requirements depend on how the business semantics or context of the data stored and processed in SAP IQ are understood.

> **Note**
>
> Using capabilities to communicate with other data sources, SAP IQ may also be used to process data that is stored in other systems and accessed through virtual tables.

In SAP IQ installations, the business semantics of data are part of the application definition and implementation. SAP IQ provides the features for working with technical database objects, such as tables. It is therefore the application that "knows," for example, which tables in the database contain sensitive personal data, or how business level objects, such as sales orders, are mapped to technical objects in the database. Applications built on top of SAP IQ need to make use of features provided by SAP IQ to implement compliance requirements for their specific use case.

SAP IQ provides a variety of security-related features to implement general security requirements that are also required for data protection and privacy:

Aspect of Data Protection and Privacy	SAP IQ Feature	More Information

Access control	Several features in SAP IQ provide access control:
	• Authentication
	• Authorization
	• Data encryption:
	  • Data volume encryption
	  • Redo log encryption
	  • Backup encryption
		See:
		• Roles [page 87]
		• Privileges [page 48]
		• Users [page 11]
		• Login Policies [page 136]
		• Data Security [page 179]

Aspect of Data Protection and Privacy	SAP IQ Feature	More Information
Access logging	Audit logging	See: • *Database activity mode* in *SAP IQ Administration: Database*
Transmission control/ communication security	Support for encrypted communication on all internal and external channels	See: • Data Confidentiality [page 155]
Availability control	Several features in SAP IQ provide availability control: • Backup and recovery • System replication (SQL Remote and MobiLink)	See: • *Backup and Recovery* in *SAP IQ Administration: Backup, Restore, and Data Recovery* • *Overview of data exchange technologies* in *SQL Anywhere Introduction*
Separation by purpose	Separation by purpose is subject to the organizational model implemented and must be applied as part of the authorization concept. Isolated data storage can be achieved in SAP IQ using: • Database schemas protected using authorization	See: • Roles [page 87] • Privileges [page 48]

> ⚠ Caution
>
> The extent to which data protection is ensured depends on secure system operation. Network security, security note implementation, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation.

In this section:

9.1 Deletion of Personal Data

SAP IQ supports the deletion of data in tables using SQL deletion commands. Applications running on SAP IQ must make use of such commands to implement deletion requirements of personal data.

End of purpose checks, including implementation of legally required retention periods and data blocking, are managed by the application. Applications can implement data blocking using SAP IQ mechanisms such as

authorization and table creation. For example, an application could transfer blocked data to separate database tables that are protected by special authorizations.

Once data has been deleted, the delete operation cannot be undone using SQL statements.

> **Note**
>
> Following standard practice, deletion of personal data is not enforced in backups. Common practice is that deleted data disappears from backups following typical backup-rotation mechanisms. SAP IQ supports backup lifecycle management by providing functions for deleting backups according to time stamps.

10 SySAM License Server Security

Safeguard the SySAM license server against vulnerabilities.

- Make sure the SySAM license server ports are secured (not open to a public network).
- Run the SySAM license server as a user with least privileges.

Related Information

SySAM Users Guide

11 External Authentication

11.1 LDAP User Authentication with SAP IQ

You can integrate SAP IQ into any existing enterprise-wide directory access framework based on Lightweight Directory Access Protocol (LDAP), a widely accepted international standard.

Integration of SAP IQ with LDAP user authentication supports:

- Authentication using searched distinguished name (DN)
- Failover to a secondary LDAP server for high availability
- Automatic failback to previously failed servers
- Integration with OpenLDAP third-party libraries
- Secure communication with LDAP servers
- Efficient design for frequent, short-lived connections
- Extensibility to multiple domains and multiple LDAP servers

In this section:

Displaying Current State for an LDAP Server Configuration Object [page 213]
Run the `sa_get_ldapserver_status` stored procedure to generate a report on the current state of an LDAP server configuration object.

Tutorial: Creating an LDAP User Authentication Environment [page 214]
This tutorial guides you through the basic steps for setting up an LDAP user authentication environment in Interactive SQL.

11.1.1 License Requirements for LDAP User Authentication

You can integrate SAP IQ into any existing enterprise-wide directory access framework based on Lightweight Directory Access Protocol (LDAP), a widely accepted international standard.

LDAP user authentication is part of the separately licensed SAP IQ Advanced Security Option (IQ_SECURITY). This option protects your environment against unauthorized access, and is required to use LDAP user authentication with SAP IQ.

11.1.2 About the LDAP Server Configuration Object

SAP IQ uses a configuration object called LDAP server to allow LDAP user authentication.

Despite its name, the LDAP server is a configuration object that resides on the SAP IQ server, rather than an actual server. Its sole function is to provide a connection to a physical LDAP server to allow LDAP user authentication. Any configuration of the LDAP server configuration object applies only to the SAP IQ side of the LDAP user authentication equation. LDAP server configuration object configuration settings are never written to the physical LDAP server.

> **Note**
>
> For the purposes of clarity in this documentation, LDAP server configuration object refers to the SAP IQ internal configuration object. LDAP server refers to the external entity.

11.1.3 Failover Capabilities When Using LDAP User Authentication

To support failover functionality, you can create a primary and a secondary LDAP server configuration object.

Each LDAP server configuration object connects to a single LDAP server and can be designated as a primary or secondary server. In the event the designated primary LDAP server configuration object cannot connect to the LDAP server, the designated secondary LDAP server configuration object is used for user authentication. You can manually manage failover and fail back using with SQL statements or be performed automatically by SAP IQ when it detects a change is appropriate.

Define primary and secondary LDAP server configuration objects in the login policy. For failover to occur, you must define both a primary and a secondary LDAP server configuration object. If only a primary LDAP

server configuration object is defined in a login policy, failover does not occur. If a secondary LDAP server configuration object is defined with no primary LDAP server configuration object, the secondary LDAP server configuration object behaves as the primary LDAP server configuration object, and failover does not occur.

When designating the secondary LDAP server configuration object, you must configure the LDAP server configuration object to connect to the correct failover LDAP server. In the event of a failover, if the secondary LDAP server configuration object cannot connect to the secondary LDAP server, LDAP user authentication in SAP IQ will be unavailable.

11.1.4 Enabling LDAP User Authentication

Configure LDAP user authentication with SAP IQ. Once configuration is complete verify that users can log on using LDAP user authentication.

1. Configuring LDAP User Authentication as a Login Method [page 197]
   To enable LDAP user authentication, you must add the value **LDAPUA** to the LOGIN_MODE database option.

11.1.4.1 Configuring LDAP User Authentication as a Login Method

To enable LDAP user authentication, you must add the value **LDAPUA** to the LOGIN_MODE database option.

Prerequisites

Requires the SET ANY SECURITY OPTION system privilege.

Context

Once set, LDAP user authentication is immediately available.

Procedure

To add the LDAPUA value to the LOGIN_MODE option, execute:

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA';
```

Task overview: Enabling LDAP User Authentication [page 197]

Related Information

LOGIN_MODE Option

11.1.5 Managing the LDAP Server Configuration Object with SAP IQ

Management includes the creation, modification and option maintenance of the LDAP server configuration object to facilitate LDAP user authentication.

In this section:

Allowing Standard Authentication in an LDAP User Authentication Only Environment [page 199]
: Allow select users to authenticate using standard authentication in an environment that supports only LDAP user authentication.

Setting the TLS Connection Trusted Relationship [page 200]
: Define the location and file name that contains the trusted relationship to be used for the Transport Layer Security (TLS) connections to the external LDAP server for user authentication.

Creating an LDAP Server Configuration Object [page 201]
: Create a new LDAP server configuration object to allow LDAP user authentication.

Validating an LDAP Server Configuration Object [page 203]
: Validate the attribute of a new or existing LDAP server configuration object.

Activating an LDAP Server Configuration Object [page 205]
: Activate an LDAP server configuration object by setting the connection state to READY. This enables LDAP user authentication.

Editing LDAP Server Configuration Object Attributes [page 205]
: Modify the existing attributes on an LDAP server. Any changes to the attributes are applied on subsequent connections. Any connection already open when the change is applied does not immediately reflect the change.

Refreshing an LDAP Server Configuration Object [page 207]
: Reinitialize the LDAP server. The command fails if the connection state of the LDAP server is not in an ACTIVE or READY state.

Suspending an LDAP Server Configuration Object [page 208]
: Put an LDAP server into maintenance mode. All connections to the LDAP server are closed and LDAP user authentication is no longer available.

Deleting an LDAP Server Configuration Object [page 208]
: Delete an LDAP server configuration object that is not in a READY or ACTIVE state.

LDAP Server Configuration Object States [page 209]
: List of possible states of an LDAP server configuration object.

Enabling Secure LDAP [page 210]

Secure LDAP uses TLS certificate authentication to provide protection against spoofing.

Syntax and Parameters for the LDAP Server Configuration Object URL [page 210]
The URL identifies the host (by name or by IP address), port number, and search to be performed when executing a secure distinguished name (DN) lookup to the LDAP server.

11.1.5.1 Allowing Standard Authentication in an LDAP User Authentication Only Environment

Allow select users to authenticate using standard authentication in an environment that supports only LDAP user authentication.

Context

If LDAP user authentication is the only authentication method allowed to access the SAP IQ database, these circumstances may create a scenario in which no user is permitted to log on:

- If no login policy exists with LDAP user authentication enabled;
- If no users are assigned to a login policy with LDAP user authorization enabled; or
- If all user accounts assigned to a login policy with LDAP user authentication are locked.

You may not be able to prevent this scenario; however, there is a method that allows a select number of users to log in to SAP IQ database using standard authentication. This method is intended as a temporary solution when LOGIN_MODE configuration prevents all users from connecting to the database.

When granting the select users access using standard authentication, ensure that at least one of those users has the SET ANY SECURITY OPTION or MANAGE ANY LOGIN POLICY system privileges to allow them to permanently resolve the issue. Depending on the underlying cause of the inability of any users to log in using LDAP user authentication, one or both of these system privileges might be required to permanently resolve the issue. You can specify a maximum of five user IDs, separated by semicolons, and enclosed in double quotation marks.

Grant standard authentication access only after the lockdown problem has occurred; you don't need to set it in advance. It does not need to be set in advance. To allow select users to log in using standard authentication, execute the `start_iq` utility with the `-al <user-id-list>` command line switch. Once granted, at the credentials prompt, the user enters his or her standard authentication user name and password.

Include the `-al` switch at either the server or database level. At the server level, the `-al` switch remains in effect until the next time the server is restarted. At the database level, the `-al` switch remains in effect until the next time the database is stopped and restarted.

Procedure

To allow standard authentication, execute one of these commands:

Level	Statement
Server	`start_iq -al <"user1,user2,user3" server_name.cfg database-name.db >`
Database	`start_iq <servername.cfg database_name.db> -al <"user1,user2,user3">`

Example

This example assumes that login_mode is set to "LDAPUA". This command allows users `Alice`, `Bob`, and `Carol` to authenticate using standard authentication on `database1` on `server1`:

```
start_iq –al  "alice;bob;carol" server1.cfg database1.db
```

Related Information

-al Database Server Option
-al Database Option

11.1.5.2 Setting the TLS Connection Trusted Relationship

Define the location and file name that contains the trusted relationship to be used for the Transport Layer Security (TLS) connections to the external LDAP server for user authentication.

Prerequisites

Requires the SET ANY SECURITY OPTION system privilege.

Context

During LDAP user authentication, SAP IQ acts as a client to the LDAP server, and must have access to the file that contains the name of the certificate authority (CA) that signed the TLS certificate. The path and file name to the CA are stored in the public-only TRUSTED_CERTIFICATES_FILE database security option. By default, this option is set to NULL (disabled), meaning that no outbound connections can be started because there are no trusted CA. Once set, this value takes effect immediately.

The list of trusted CAs that sign server certificates may be shared in a location in a Windows environment on the local C: drive for all SAP applications on that machine.

Procedure

To set the TRUSTED_CERTIFICATES_FILE database security option, execute:

```
SET OPTION PUBLIC.TRUSTED_CERTIFICATES_FILE = '<path>/<filename'>
```

Example

This example sets the path to the trusted certificates file to `C:\sap\shared`, in a file called `\trusted.txt`:

```
SET OPTION PUBLIC.TRUSTED_CERTIFICATES_FILE = 'C:\sap\shared\trusted.txt'
```

11.1.5.3 Creating an LDAP Server Configuration Object

Create a new LDAP server configuration object to allow LDAP user authentication.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Context

The LDAP server configuration object provides a connection between SAP IQ and a physical LDAP server. If you are using multiple LDAP servers, particularly for failover, set up a separate LDAP server configuration object for each LDAP server. The parameters of the LDAP server configuration object are stored in the `ISYSLDAPSERVER` (system view SYSLDAPSERVER) system table. To automatically activate the connection to the LDAP server upon creation, use the WITH ACTIVATE clause.

Procedure

1. Identify the values for the applicable SEARCH DN attributes to be defined for the new LDAP server configuration object.

SEARCH DN Attribute	Valid Values
URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
ACCESS ACCOUNT	The distinguished name for a user connecting to the external LDAP server.
IDENTIFIED BY	The password associated with the ACCESS ACCOUNT distinguished name.
IDENTIFIED BY ENCRYPTED	The encrypted password associated with the ACCESS ACCOUNT distinguished name.

2. Identify the values for the applicable LDAPUA server attributes for the new LDAP server configuration object.

LDAPUA Attribute	Valid Values
SEARCH DN	All attributes defined from SEARCH DN Attributes (see step 1).
AUTHENTICATION URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
CONNECTION TIMEOUT	Specifies the connection timeout value for both DN searches and authentication between SAP IQ and the external LDAP server. Specified in milliseconds, the default value is 10 seconds.
CONNECTION RETRIES	Specifies the number of retries on connections from SAP IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1 – 60, with a default value of 3.
TLS	Defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server both for DN searches and authentication. The valid settings are ON and OFF (default). **Note** See *Enabling Secure LDAP* and *Setting the TLS Connection Trusted Relationship*.

3. Execute the `CREATE LDAP SERVER` statement, specifying the applicable attributes and clauses. For example:

```
CREATE LDAP SERVER secure_primary
SEARCH DN
     URL 'ldaps://my_LDAPserver:636/dc=MyCompany,dc=com??sub?cn=*'
     ACCESS ACCOUNT 'cn=myadmin, cn=Users, dc=mycompany, dc=com'
     IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldaps://my_LDAPserver:636/'
CONNECTION TIMEOUT 3000
CONNECTION RETRIES 3
```

```
TLS OFF
WITH ACTIVATE
```

11.1.5.4 Validating an LDAP Server Configuration Object

Validate the attribute of a new or existing LDAP server configuration object.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Context

The `VALIDATE LDAP SERVER` command is useful for an administrator when setting up a new LDAP server configuration object or when diagnosing connection issues between SAP IQ and the LDAP server. Any connection established by the VALIDATE LDAP SERVER statement is temporary and closed at the end of the execution of the statement.

To validate the existence of the user on the LDAP server, include the `CHECK` clause. Specify the userID and the `<user-dn-string>` to be compared.

Procedure

1. Identify the SEARCH DN attributes of the LDAP server configuration object to be validated.

SEARCH DN Attribute	Valid Values
URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
ACCESS ACCOUNT	The distinguished name for a user connecting to the external LDAP server.
IDENTIFIED BY	The password associated with the ACCESS ACCOUNT distinguished name.
IDENTIFIED BY ENCRYPTED	The encrypted password associated with the ACCESS ACCOUNT distinguished name.

2. Identify the LDAPUA attributes of the LDAP server configuration object to be validated.

LDAPUA Attribute	Valid Values
SEARCH DN	All attributes defined from SEARCH DN Attributes (see step 1).
AUTHENTICATION URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
CONNECTION TIMEOUT	Specifies the connection timeout value for both DN searches and authentication between SAP IQ and the external LDAP server. Specified in milliseconds, the default value is 10 seconds.
CONNECTION RETRIES	Specifies the number of retries on connections from SAP IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1 – 60, with a default value of 3.
TLS	Defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server both for DN searches and authentication. The valid settings are ON and OFF (default). **Note** See *Enabling Secure LDAP* and *Setting the TLS Connection Trusted Relationship*.

3. Execute the `VALIDATE LDAP SERVER` command with the applicable attributes.

Example

For example, assume the LDAP server configuration object named `apps_primary` was created as follows and the `SET OPTION PUBLIC.login_mode` is set to `'Standard,LDAPUA'`:

```
CREATE LDAP SERVER apps_primary
SEARCH DN
    URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'
    ACCESS ACCOUNT 'cn=myadmin, cn=Users, dc=mycompany, dc=com'
    IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'
CONNECTION TIMEOUT 3000
WITH ACTIVATE
```

This statement validates the existence of a userID `myusername` by comparing it to the expected user distinguished name (enclosed in quotation marks) on the LDAP server configuration object name `apps_primary` using the optional CHECK clause:

```
VALIDATE LDAP SERVER apps_primary
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com'
```

11.1.5.5 Activating an LDAP Server Configuration Object

Activate an LDAP server configuration object by setting the connection state to READY. This enables LDAP user authentication.

Prerequisites

For SAP IQ 16.0 and higher, requires the MANAGE ANY LDAP SERVER system privilege.

Context

LDAP server configuration object attribute values are read from the `ISYSLDAPSERVER` system table and applied to new connections to the LDAP server and incoming authentication requests to the SAP IQ server. Upon successful authentication of a user, the connection state to the LDAP server changes to ACTIVE.

Procedure

To activate an LDAP server configuration object, execute:

```
ALTER LDAP SERVER <LDAP_server_name> WITH ACTIVATE
```

11.1.5.6 Editing LDAP Server Configuration Object Attributes

Modify the existing attributes on an LDAP server. Any changes to the attributes are applied on subsequent connections. Any connection already open when the change is applied does not immediately reflect the change.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Procedure

1. Identify the existing SEARCH DN attributes to be modified.

SEARCH DN Attribute	Valid Values
URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
ACCESS ACCOUNT	The distinguished name for a user connecting to the external LDAP server.
IDENTIFIED BY	The password associated with the ACCESS ACCOUNT distinguished name.
IDENTIFIED BY ENCRYPTED	The encrypted password associated with the ACCESS ACCOUNT distinguished name.

2. Identify the existing LDAPUA attributes to be modified.

LDAPUA Attribute	Valid Values
SEARCH DN	All attributes defined from SEARCH DN Attributes (see step 1).
AUTHENTICATION URL	Specify the host (by name or by IP address), port number, and search to be performed to lookup the DN for a given user ID or enter NULL. **Note** See *Syntax and Parameters for the LDAP Server Configuration Object URL* for supported syntax.
CONNECTION TIMEOUT	Specifies the connection timeout value for both DN searches and authentication between SAP IQ and the external LDAP server. Specified in milliseconds, the default value is 10 seconds.
CONNECTION RETRIES	Specifies the number of retries on connections from SAP IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1 – 60, with a default value of 3.
TLS	Defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server both for DN searches and authentication. The valid settings are ON and OFF (default). **Note** See *Enabling Secure LDAP* and *Setting the TLS Connection Trusted Relationship*.

3. Identify the server clauses to be used.

Clause	Description
WITH SUSPEND	Puts the LDAP server into maintenance mode

Clause	Description
WITH ACTIVATE	Puts the LDAP server in a READY state and enables LDAP authentication
WITH REFRESH	Reinitializes LDAP user authentication

4. Execute the `ALTER LDAP SERVER` command with the applicable parameters and clauses, for example:

Example

```
ALTER LDAP SERVER apps_primary
AUTHENTICATION URL 'ldap://my_LDAPserver:1066/'
CONNECTION RETRIES 10
WITH ACTIVATE
```

11.1.5.7 Refreshing an LDAP Server Configuration Object

Reinitialize the LDAP server. The command fails if the connection state of the LDAP server is not in an ACTIVE or READY state.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Context

When refreshing an LDAP server, all connections to the LDAP server are closed and the option values on the LDAP server are reread from the `ISYSLDAPSERVER` system table. The values are then applied to all new connections to the LDAP server and all incoming user authentication requests to the SAP IQ server. Execution of the REFRESH command does not change the connection state of the LDAP server, nor does it change any existing connections from a client to the SAP IQ server.

To ensure that any changes are used when a user next authenticates, it is recommended that you refresh the LDAP server after making any changes to the TRUSTED_CERTIFICATES_FILE database option or to the contents of the file specified by the TRUSTED_CERTIFICATES_FILE database option.

Procedure

To refresh the LDAP server, execute:

```
ALTER LDAP SERVER <LDAP_server_name>
WITH REFRESH
```

11.1.5.8 Suspending an LDAP Server Configuration Object

Put an LDAP server into maintenance mode. All connections to the LDAP server are closed and LDAP user authentication is no longer available.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Procedure

To suspend an LDAP server, execute:

```
ALTER LDAP SERVER <LDAP_server_name>
WITH SUSPEND
```

Related Information

ALTER LDAP SERVER Statement
LDAP Server Configuration Object States [page 209]

11.1.5.9 Deleting an LDAP Server Configuration Object

Delete an LDAP server configuration object that is not in a READY or ACTIVE state.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Context

The DROP statement fails when it is issued against an LDAP server configuration object that is in a READY or ACTIVE state. The DROP statement also fails if a login policy exists with a reference to the LDAP server configuration object being dropped. To ensure any references to the LDAP server configuration object are removed from all login policies before being dropped, include the WITH DROP ALL REFERENCES clause. To override the server state check and put the database object into maintenance mode regardless of its current state, include the WITH SUSPEND clause when dropping an LDAP server configuration object.

Dropping an LDAP server configuration object removes the named object from the `ISYSLDAPSERVER` system table.

Procedure

To drop an LDAP server configuration object, execute this command, including the applicable clauses:

```
DROP LDAP SERVER <LDAP_Server_name> WITH SUSPEND WITH DROP ALL REFERENCES
```

Example

This example drops the LDAP server configuration object named `ldapserver1` regardless of its current state and removes any references to `ldapserver1` in all login policies:

```
DROP LDAP SERVER ldapserver1 WITH DROP ALL REFERENCES WITH SUSPEND
```

This `DROP LDAP SERVER` command fails if the LDAP server configuration object named `ldapserver2` is referenced in any login policies because the WITH DROP ALL REFERENCES clause is not included:

```
DROP LDAP SERVER ldapserver1 WITH SUSPEND
```

Related Information

DROP LDAP SERVER Statement
LDAP Server Configuration Object States [page 209]

11.1.5.10 LDAP Server Configuration Object States

List of possible states of an LDAP server configuration object.

The state of an LDAP server configuration object is maintained persistently on writeable databases in the `ISYSLDAPSERVER` system table to provide visibility for administrators into LDAP user authentication. If an

LDAP server configuration object is restarted, the state at the time of shutdown is retained. This permits maintenance on an LDAP server configuration object to remain in force throughout restarts. With read-only databases, state changes are not stored persistently – they occur only in memory, and are lost when the database is shut down. The connection state is set at start-up using the value from a read-only database, and transient state changes may occur in memory to provide LDAP user authentication.

The possible states of an LDAP server configuration object include:

RESET one or more attributes on the LDAP server configuration object have been entered or modified since last activation.

READY the LDAP server configuration object is ready to accept connections.

ACTIVE the LDAP server configuration object has performed at least one successful LDAP user authentication.

FAILED there is a problem connecting to the LDAP server configuration object.

SUSPENDED the LDAP server configuration object is in maintenance mode, and is unavailable for LDAP user authentication.

11.1.5.11 Enabling Secure LDAP

Secure LDAP uses TLS certificate authentication to provide protection against spoofing.

Use of a TLS certificate provides the client connection to the LDAP server with proof that the server is who it says it is.

Enabling Secure LDAP on an LDAP server configuration object can take one of two forms:

ldaps:// on the LDAP server configuration object, use ldaps:// when defining the SEARCH DN URL or AUTHENTICATION URL attributes and set the TLS attribute to OFF.

TLS parameter on the LDAP server configuration object, use ldap:// when defining the SEARCH DN URL attribute and set the TLS attribute to ON.

> ⓘ Note
>
> Current versions of Active Directory (AD), Tivoli, SunONE Oracle DS, and OpenLDAP support both options. Older versions may only support one option. For compatibility with all versions, both options are supported by SAP IQ.

11.1.5.12 Syntax and Parameters for the LDAP Server Configuration Object URL

The URL identifies the host (by name or by IP address), port number, and search to be performed when executing a secure distinguished name (DN) lookup to the LDAP server.

While the syntax of the URL can take one of two forms depending on how the secure connection to the LDAP server is to be made, the underlying parameters of the URL are the same for each form.

ldaps:// on the LDAP server configuration object, use ldaps:// when defining the SEARCH DN URL or AUTHENTICATION URL attributes and set the TLS attribute to OFF.

```
ldapurl::=ldaps://host:[port]/[node]?[attributes]? [base | one | sub]?
[filter]
```

TLS parameter on the LDAP server configuration object, use ldap:// when defining the SEARCH DN URL attribute and set the TLS attribute to ON.

```
ldapurl::=ldap://host:[port]/[node]?[attributes]? [base | one | sub]? [filter]
```

Parameter	Description
host	The host name of the LDAP server.
port	The port number of the LDAP server. f no port is specified, the standard LDAP ports, 389 for LDAP and 636 for secure LDAP, are used.
node	The node in the object hierarchy at which to start the search.
attributes	A list of attributes returned in the result set. Each LDAP server may support a different attribute based on the schemas used by the LDAP server. However, for each LDAP server, only the first attribute is used and should return the distinguished name (DN) of the user. If an attribute is not specified, then the default "entryDN" is used. If an unrecognized attribute is specified, then no attributes is returned.
base | one | sub	Qualifies the search criteria. base – Specifies a search of the base node. one – Specifies a search of node and one sublevel. sub – Specifies a search of node and all sublevels. If no scope is specified, the server performs a base search. The sub scope is recommended.
filter	Specifies the attribute or attributes used to search for a database user's distinguished name (DN). The filter can be simple, such as "uid=*," or compound, such as "(uid=*) (ou=group)." The attributes in the filter are dependent on the LDAP server schema. LDAP user authentication replaces each wildcard character (*) with the database user ID when searching for a DN.

The URL is initially defined as one of the server attributes when creating an LDAP server configuration object and can be changed at any time. If there is no default, you can omit parameters by using two consecutive question marks (for example, /dc=sap,dc=com??sub?uid=*). Creating or modifying the LDAP server configuration object requires the MANAGE ANY LDAP SERVER system privilege.

> **Note**
>
> Current versions of Active Directory (AD), Tivoli, SunONE Oracle DS, and OpenLDAP support both options. Older versions may only support one option. For compatibility with all versions, both options are supported by SAP IQ.

11.1.6 Managing LDAP User Authentication Login Policy Options

There are several login policy options specific to LDAP user authentication. These options must be defined in a login policy assigned to a user using LDAP user authentication.

The steps for creating, managing, and assigning a login policy to a user are the same as for regular login policies.

See Login Policies [page 136]

Option	Description	Properties
LDAP_AUTO_FAIL-BACK_PERIOD	Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.	Values – 0–2147483647 Default – 15 minutes Applies to all users
LDAP_FAIL-OVER_TO_STD	Permits authentication with standard authentication when authentication with the LDAP server fails due to system resources, network outage, connection timeouts, or similar system failures. However, it does not permit an actual authentication failure returned from an LDAP server to fail over to standard authentication.	Values – ON; OFF Default – ON Applies to all users
LDAP_PRI-MARY_SERVER	Specifies the name of the primary LDAP server.	Values – N/A Default – none Applies to all users
LDAP_REFRESH_DN	Updates the ldap_refresh_dn value in the ISYSLOGINPOLICYOPTION system table with the current time, stored in Coordinated Universal Time (UTC). Each time a user authenticates with LDAP, if the value of ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the value of user_dn in ISYSUSER, a search for a new user DN occurs. The user_dn value is then updated with the new user DN and the user_dn_changed_at value is again updated to the current time.	Values – NOW Initial value for ROOT policy – NULL Initial value for user-defined login policy – current time stored in UTC
LDAP_SECON-DARY_SERVER	Specifies the name of the secondary LDAP server.	Values – N/A Default – none Applies to all users

11.1.7 Manage Users and Passwords with LDAP User Authentication

To log in to SAP IQ using LDAP user authentication, each user must have an active user ID and password on the external LDAP server as well as an active user ID on the SAP IQ server.

When creating a new user in SAP IQ, though not required, it is recommended that you specify a password to ensure that the new user account is not left unprotected until the first LDAP user authentication login.

The first time a new user logs on or an existing user logs in after a password change, the password in the SAP IQ database is automatically overwritten with the corresponding user password defined on the external LDAP

server. Therefore, all maintenance required on SAP IQ passwords for user using LDAP user authentication should always be done on the external LDAP server, not the SAP IQ server.

As a result of this automatic password synchronization, for users granted the ability to use Standard authentication (the password defined in the SAP IQ database), when attempting to log on when using Standard authentication, they should continue to use their LDAP server credentials.

11.1.8 Displaying Current Status Information for a User

Run the `sa_get_user_status` stored procedure to generate a report about the current status of a user.

Information includes connection and failed login information as well as whether the user has been locked out and if so, why. If the user is authenticated using LDAP user authentication, the output includes the user's distinguished name and the date and time that the distinguished name was found.

The MANAGE ANY USER system privilege is required to run this stored procedure. A user without the MANAGE ANY USER system privilege can obtain user information by creating and executing a cover procedure owned by a user with MANAGE ANY USER system privilege.

Related Information

sa_get_user_status System Procedure

11.1.9 Displaying Current State for an LDAP Server Configuration Object

Run the `sa_get_ldapserver_status` stored procedure to generate a report on the current state of an LDAP server configuration object.

Status information includes the LDAP server configuration object name, object identifier, current state, and the date and time of the last state change. A properly configured and running LDAP server configuration object has a state of READY or ACTIVE.

No system privilege is required to run this stored procedure.

Related Information

sa_get_ldapserver_status System Procedure

11.1.10 Tutorial: Creating an LDAP User Authentication Environment

This tutorial guides you through the basic steps for setting up an LDAP user authentication environment in Interactive SQL.

1. Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL [page 214]
   Use Interactive SQL to create a server configuration object
2. Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 216]
   Use Interactive SQL to create a login policy that uses the LDAP server.
3. Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL [page 217]
   Use Interactive SQL to create users that authenticate to the LDAP server by using that login policy.

11.1.10.1 Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL

Use Interactive SQL to create a server configuration object

Prerequisites

You must have an LDAP server.

You must have completed the previous lessons in this tutorial.

You must have the system privileges listed at the beginning of this tutorial.

Procedure

1. Enable TLS encryption for communications with your LDAP server.
   a. Specify a certificate file for the server to use. For example:

      ```
      SET OPTION PUBLIC.trusted_certificates_file='c:\\certificates\
      \trusted.txt';
      ```

   b. If you do not have a certificate, you can try this tutorial without TLS encryption. Unencrypted communication with an LDAP server (plain text exchanges) is not recommended since you risk exposing your authentication credentials including password. You should only attempt this when the LDAP server is inside the same firewall as the database server. To continue without TLS, replace all TLS ON clauses with TLS OFF in this tutorial.
2. Execute a VALIDATE LDAP SERVER statement to test the connection to an LDAP server.

For example, the following statement verifies the connection attributes of an existing LDAP server. The database server connects to the LDAP server with the supplied credentials.

```
VALIDATE LDAP SERVER
    SEARCH DN
        URL 'ldap://iq10web:389/dc=sap,dc=com?dn?sub?uid=*'
        ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
        IDENTIFIED BY 'Not4YourEyes'
    AUTHENTICATION URL 'ldap://iq10web:389/'
    CONNECTION TIMEOUT 1000
    CONNECTION RETRIES 3
    TLS ON;
```

This LDAP server uses port 389 for communications. Your VALIDATE LDAP SERVER statement must execute without error before continuing to the next step.

3. Execute a CREATE LDAP SERVER statement to create an LDAP server configuration object.

   For example, the following statement defines an LDAP server configuration object that can be used for user authentication.

   ```
   CREATE LDAP SERVER prim_ldap
       SEARCH DN
           URL 'ldap://iq10web:389/dc=sap,dc=com?dn?sub?uid=*'
           ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
           IDENTIFIED BY 'Not4YourEyes'
       AUTHENTICATION URL 'ldap://iq10web:389/'
       CONNECTION TIMEOUT 1000
       CONNECTION RETRIES 3
       TLS ON;
   ```

   Unlike the VALIDATE LDAP SERVER statement, the CREATE LDAP SERVER statement does not attempt a connection to the LDAP server.

4. Execute a CREATE LDAP SERVER statement to create a second LDAP server configuration object to be used for failover. This step is optional but is required for the steps that follow.

   For example, the following statement defines an LDAP server configuration object that can be used as a failover for user authentication.

   ```
   CREATE LDAP SERVER sec_ldap
       SEARCH DN
           URL 'ldap://iq10web:390/dc=sap,dc=com?dn?sub?uid=*'
           ACCESS ACCOUNT 'cn=Manager,dc=sap,dc=com'
           IDENTIFIED BY 'Not4YourEyes'
       AUTHENTICATION URL 'ldap://iq10web:390/'
       CONNECTION TIMEOUT 1000
       CONNECTION RETRIES 3
       TLS ON;
   ```

   This LDAP server uses port 390 for communications.

5. Execute a SET TEMPORARY OPTION statement to change the login mode to enable LDAP user authentication. For additional security, use SET TEMPORARY OPTION to set the login_mode option. If the database is copied, then this setting prevents unauthorized access from a spoofed LDAP server.

   ```
   SET TEMPORARY OPTION PUBLIC.login_mode='Standard,LDAPUA';
   ```

   The current setting of the login_mode option can be queried as follows:

   ```
   SELECT connection_property('login_mode');
   ```

Results

An LDAP server configuration object is created, and references to it are added to the ISYSLDAPSERVER system table.

Using LDAPS:// and no TLS clause is equivalent to LDAP:// with the TLS ON clause.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating an LDAP User Authentication Environment [page 214]

Next task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 216]

11.1.10.2 Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL

Use Interactive SQL to create a login policy that uses the LDAP server.

Procedure

Execute a CREATE LOGIN POLICY statement to create a new login policy.

For example, the following statement creates a new login policy that can be used to authenticate users.

```
CREATE LOGIN POLICY ldap_policy_both
  LDAP_PRIMARY_SERVER=prim_ldap
  LDAP_SECONDARY_SERVER=sec_ldap
  LDAP_FAILOVER_TO_STD=ON;
```

The names of the primary and secondary LDAP servers are specified in this login policy. If authentication for the user associated with this login policy fails, then standard authentication is attempted (if it is permitted by the login mode). The current settings of the login policy can be queried as follows:

```
SELECT lpo.* FROM SYS.SYSLOGINPOLICYOPTION AS lpo,
    SYS.SYSLOGINPOLICY AS lp
  WHERE lpo.login_policy_id = lp.login_policy_id
    AND lp.login_policy_name = 'ldap_policy_both';
```

Results

A login policy is created that uses an LDAP server for user authentication and definitions for it are added to the SYSLOGINPOLICY and SYSLOGINPOLICYOPTION system tables.

Next Steps

Proceed to the next lesson.

Task overview: Tutorial: Creating an LDAP User Authentication Environment [page 214]

Previous task: Lesson 1: Creating an LDAP Server Configuration Object Using Interactive SQL [page 214]

Next task: Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL [page 217]

11.1.10.3 Lesson 3: Creating a User That Authenticates to an LDAP Server Using Interactive SQL

Use Interactive SQL to create users that authenticate to the LDAP server by using that login policy.

Procedure

1. Execute a CREATE USER statement to create a new user ID with the LDAP login policy defined in a previous step.

   For example, the following statement creates a new user ID that authenticates against either the primary or secondary LDAP server.

   ```
   CREATE USER ldap_user01 LOGIN POLICY ldap_policy_both;
   ```

   The IDENTIFIED BY clause is omitted. The IDENTIFIED BY clause then specifies the password to use for standard authentication. If a password is specified using this clause, it need not be the same as the password authenticated by the LDAP server. Since this password is replaced the first time the user successfully authenticates to the LDAP server, the IDENTIFIED BY clause is omitted here.
2. Activate the LDAP servers for immediate use. The following statements activate the primary and secondary LDAP servers.

   ```
   ALTER LDAP SERVER prim_ldap WITH ACTIVATE;
   ALTER LDAP SERVER sec_ldap WITH ACTIVATE;
   ```

The following query determines the current state of all LDAP servers.

```
CALL sa_get_ldapserver_status;
```

The ldsrv_state column of the result set indicates that the two LDAP servers are in the READY state.

3. Execute an Interactive SQL CONNECT statement to connect to the database with LDAP user authentication.

   For example, the following statement connects to the sample database by using the specified user ID and password.

   ```
   CONNECT DATABASE demo USER ldap_user01 IDENTIFIED BY 'abcd1234';
   ```

   If the LDAP server fails the user authentication, then standard authentication is attempted. Standard authentication can fail if the standard password does not match the one provided (for example, because it has not been updated during LDAP authentication).

Results

A database user is created that uses an LDAP server to authenticate. Whenever the LDAP server is not available, standard authentication takes place.

Task overview: Tutorial: Creating an LDAP User Authentication Environment [page 214]

Previous task: Lesson 2: Creating a Login Policy That Uses an LDAP Server Using Interactive SQL [page 216]

11.2 Kerberos User Authentication

The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating system, and network logins.

The Kerberos login is more convenient for users and permits a single security system for database and network security. Its advantages include:

- The user does not need to provide a user ID or password to connect to the database.
- Multiple users can be mapped to a single database user ID.
- The name and password used to log in to Kerberos do not have to match the database user ID and password.

Kerberos is a network authentication protocol that provides strong authentication and encryption using secret-key cryptography. Users already logged in to Kerberos can connect to a database without providing a user ID or password.

Kerberos can be used for authentication. To delegate authentication to Kerberos you must:

- configure the server and database to use Kerberos logins.

- create mapping between the user ID that logs in to the computer or network, and the database user.

> ⚠ Caution
>
> When using Kerberos logins as a single security solution, be sure to inform yourself on the security concern related to copied databases.

SAP IQ does not include the Kerberos software; it must be obtained separately. The following components are included with the Kerberos software:

Kerberos libraries

These are referred to as the Kerberos Client or GSS (Generic Security Services)-API runtime library. These Kerberos libraries implement the well-defined GSS-API. The libraries are required on each client and server computer that intends to use Kerberos. The built-in Windows SSPI interface can be used instead of a third-party Kerberos client library if you are using Active Directory as your KDC.

SSPI can only be used by clients in the Kerberos connection parameter. The database server cannot use SSPI. It needs a supported Kerberos client other than SSPI.

A Kerberos Key Distribution Center (KDC) server

The KDC functions as a storehouse for users and servers. It also verifies the identification of users and servers. The KDC is typically installed on a server computer not intended for applications or user logins.

Kerberos authentication from DBLib, ODBC, OLE DB, and ADO.NET clients, and SAP Open Client and jConnect clients is supported. Kerberos authentication can be used with SAP IQ transport layer security encryption, but Kerberos encryption for network communications is not supported.

Windows uses Kerberos for Windows domains and domain accounts. Active Directory Windows Domain Controllers implement a Kerberos KDC. A third-party Kerberos client or runtime is still required on the database server computer for authentication in this environment, but the Windows client computers can use the built-in Windows SSPI interface instead of a third-party Kerberos client or runtime.

In this section:

Security: Use Login Modes to Secure the Database [page 228]
There are several measures you can take to secure your database.

11.2.1 Licensing Requirements for Kerberos

SAP IQ supports Kerberos authentication, a login feature that allows you to maintain a single user ID and password for both database connections and operating system and network logins.

You can use your Kerberos credentials to connect to the database without specifying a user ID or password.

Kerberos authentication is part of the separately licensed SAP IQ Advanced Security Option (IQ_SECURITY). This option protects your environment against unauthorized access, and is required to use Kerberos authentication with SAP IQ.

11.2.2 Kerberos Clients

Kerberos authentication is available on several platforms.

The following table lists the default names and locations of the keytab and GSS-API files used by the supported Kerberos clients.

> **Note**
>
> SSPI can only be used by clients in the Kerberos connection parameter. The database server cannot use SSPI. It needs a supported Kerberos client other than SSPI.

Kerberos client	Default keytab file	GSS-API library file name	Notes
Windows MIT Kerberos client	`C:\WINDOWS\krb5kt`	`gssapi32.dll` or `gssapi64.dll`	The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.
Windows CyberSafe Kerberos client	`C:\Program Files\CyberSafe\v5srvtab`	`gssapi32.dll` or `gssapi64.dll`	The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.
UNIX/Linux MIT Kerberos client	`/etc/krb5.keytab`	`libgssapi_krb5.so`[1]	The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.
UNIX/Linux CyberSafe Kerberos client	`/krb5/v5srvtab`	`libgss.so`[1]	The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.

Kerberos client	Default keytab file	GSS-API library file name	Notes
UNIX/Linux Heimdal Kerberos client	`/etc/krb5.keytab`	`libgssapi.so.1`[1]	

[1] These file names may vary depending on your operating system and Kerberos client version.

11.2.3 Setting up a Kerberos System

Configure Kerberos authentication to be used with SAP IQ.

Prerequisites

You must be logged in to your computer using Kerberos authentication.

Context

Kerberos is a network authentication protocol that provides strong authentication and encryption using secret-key cryptography.

Procedure

1. If necessary, install and configure the Kerberos client software, including the GSS-API runtime library, on both the client and server.

   On Windows client computers using an Active Directory Key Distribution Center (KDC), SSPI can be used and you do not need to install the Kerberos client.
2. If necessary, create a Kerberos principal in the Kerberos KDC for each user.

   A Kerberos principal is a Kerberos user ID in the format `<user>/<instance>@<REALM>`, where `/<instance>` is optional. If you are already using Kerberos, the principal should already exist, so you do not need to create a Kerberos principal for each user.

   Principals are case sensitive and must be specified in the correct case. Mappings for multiple principals that differ only in case are not supported (for example, you cannot have mappings for both jjordan@MYREALM.COM and JJordan@MYREALM.COM).
3. Create a Kerberos principal in the KDC for the SAP IQ database server.

   The default Kerberos principal for the database server has the format `<server-name>@<REALM>`, where `<server-name>` is the SAP IQ database server name. To use a different server principal, use the -kp server option. Principals are case sensitive, and `<server-name>` cannot contain multibyte characters, or the characters `/`, `\`, or `@`.

You must create a server service principal within the KDC because servers use a keytab file for KDC authentication. The keytab file is protected and encrypted.

4. Securely extract and copy the keytab for the server principal (`<server-name>@<REALM>` by default, or the principal used with -kp server option) from the KDC to the computer running the SAP IQ database server. The default location of the keytab file depends on the Kerberos client and the platform. The keytab file's permissions should be set so that the SAP IQ server can read it, but unauthorized users do not have read permission.

Results

The Kerberos system is authenticated and configured to be used with SAP IQ.

Next Steps

Configure your SAP IQ database server and database to use Kerberos.

11.2.4 Configuring SAP IQ Databases to Use Kerberos (SQL)

Configure databases to use Kerberos logins.

Prerequisites

You must have the SET ANY PUBLIC OPTION and MANAGE ANY USER system privileges.

You must already have Kerberos configured before SAP IQ can use it.

Context

The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating systems, and network logins.

Procedure

1. Start the database server with the -krb or -kr option to enable Kerberos authentication, or use the -kl option to specify the location of the GSS-API library and enable Kerberos.

2. Change the public or temporary public option login_mode to a value that includes Kerberos. As database options apply only to the database in which they are found, different databases can have a different Kerberos login setting, even if they are loaded and running on the same database server. For example:

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Kerberos,Standard';
```

> ⚠ Caution
>
> Setting the login_mode database option to Kerberos restricts connections to only those users who have been granted a Kerberos login mapping. Attempting to connect using a user ID and password generates an error unless you are a user with SYS_AUTH_DBA_ROLE compatibility role.

3. Create a database user ID for the client user. You can use an existing database user ID for the Kerberos login, as long as that user has the correct privileges. For example:

```
CREATE USER "kerberos-user"
IDENTIFIED BY abc123;
```

4. Execute a GRANT KERBEROS LOGIN TO statement to create a mapping from the client's Kerberos principal to an existing database user ID. For example:

```
GRANT KERBEROS LOGIN TO "pchin@MYREALM.COM"
AS USER "kerberos-user";
```

   To connect when a Kerberos principal is used that does not have a mapping, ensure the Guest database user ID exists and has a password.

5. Ensure the client user has already logged on (has a valid Kerberos ticket-granting ticket) using their Kerberos principal and that the client's Kerberos ticket has not expired. A Windows user logged in to a domain account already has a ticket-granting ticket, which allows them to authenticate to servers, providing their principal has enough permissions.

   A ticket-granting ticket is a Kerberos ticket encrypted with the user's password that is used by the Ticket Granting Service to verify the user's identity.

6. Connect from the client, specifying the KERBEROS connection parameter (Often KERBEROS=YES, but KERBEROS=SSPI or KERBEROS=<GSS-API-library-file> can also be used). If the user ID or password connection parameters are specified, they are ignored. For example:

```
dbisql -c "KERBEROS=YES;Server=my_server_princ"
```

Results

The database is configured to use Kerberos authentication.

Example

A connection attempt using the following SQL statement is successful if the user logs in with a user profile name that matches a Kerberos login mapping for the default database on a database server:

```
CONNECT USING 'KERBEROS=YES';
```

The CONNECT statement can connect to a database if all the following conditions are true:

- A database server is currently running.
- The default database on the current database server is enabled to accept Kerberos authenticated connections.
- A Kerberos login mapping has been created for the user's current Kerberos principal.
- If the user is prompted with a window by the database server for more connection information (such as occurs when using Interactive SQL), the user clicks **OK** without providing more information.

Next Steps

You can use Kerberos authentication to connect from a client. Optionally, you can create a Kerberos login mapping.

Related Information

Setting up a Kerberos System [page 221]
Connecting Using SSPI for Kerberos Logins on Microsoft Windows [page 225]
-kl Database Server Option
-kr Database Server Option (Deprecated)
-krb Database Server Option

11.2.5 Connections from an SAP Open Client or jConnect Application

The database server accepts connections from an SAP Open Client or jConnect application.

- Set up Kerberos user authentication.
- Configure SAP IQ to use Kerberos.
- Set up SAP Open Client or jConnect as you would for Kerberos user authentication with Adaptive Server Enterprise. The server name must be the SAP IQ server's name and is case sensitive. You cannot connect using an alternate server name from Open Client or jConnect.

Related Information

Setting up a Kerberos System [page 221]
Configuring SAP IQ Databases to Use Kerberos (SQL) [page 222]
-krb Database Server Option
-kr Database Server Option (Deprecated)
-kl Database Server Option
Troubleshooting: Kerberos Connections [page 226]

11.2.6 Connecting Using SSPI for Kerberos Logins on Microsoft Windows

Connect using SSPI without a Kerberos client installed on the client computer.

Prerequisites

You must already have Kerberos configured before SAP IQ can use it. You must already have your database server and database configured to use Kerberos.

Context

In a Windows domain, SSPI can be used on Windows-based computers without a Kerberos client installed on the client computer. Windows domain accounts already have associated Kerberos principals.

SSPI can only be used by SAP IQ clients in the Kerberos connection parameter. SAP IQ database servers cannot use SSPI. They need a supported Kerberos client other than SSPI.

Procedure

Connect to the database from the client computer. For example:

```
dbisql -c "KERBEROS=SSPI;Server=my_server_princ"
```

When Kerberos=SSPI is specified in the connection string, a Kerberos login is attempted.

A connection attempt using the following SQL statement also succeeds, providing the user has logged on with a user profile name that matches a Kerberos login mapping for the default database on a database server:

```
CONNECT USING 'KERBEROS=SSPI';
```

Results

You can use SSPI for Kerberos authentication on Windows.

11.2.7 Troubleshooting: Kerberos Connections

If you get unexpected errors when attempting to enable or use Kerberos authentication, it is recommended that you enable additional diagnostic messages on the database server and client.

Specifying the -z option when you start the database server, or using `CALL sa_server_option( 'DebuggingInformation', 'ON' )` if the server is already running includes additional diagnostic messages in the database server message log. The LogFile connection parameter writes client diagnostic messages to the specified file.

As an alternative to using the LogFile connection parameter, you can run the Ping utility (dbping) with the -z parameter. The -z parameter displays diagnostic messages that should help identify the cause of the connection problem.

Difficulties Starting the Database Server

Symptom	Common solutions
"Unable to load Kerberos GSS-API library" message	• Ensure a Kerberos client is installed on the database server computer, including the GSS-API library. • The database server -z output lists the name of the library that it is attempting to load. Verify the library name is correct. If necessary, use the -kl option to specify the correct library name. • Ensure the directory and any supporting libraries is listed in the library path (%PATH% on Windows). • If the database server -z output states the GSS-API library was missing entry points, then the library is not a supported Kerberos Version 5 GSS-API library.

Symptom	Common solutions
"Unable to acquire Kerberos credentials for server name "<server-name>"" message	• Ensure there is a principal for <server-name>@<REALM> in the KDC. Principals are case sensitive, so ensure the database server name is in the same case as the user portion of the principal name. • Ensure the name of the SAP IQ server is the primary/user portion of the principal. • Ensure that the server's principal has been extracted to a keytab file and the keytab file is in the correct location for the Kerberos client. • If the default realm for the Kerberos client on the database server computer is different from the realm in the server principal, use the -kr option to specify the realm in the server principal.
"Kerberos login failed" client error	• Check the database server diagnostic messages. Some problems with the keytab file used by the server are not detected until a client attempts to authenticate.

Troubleshooting Kerberos Client Connections

If the client got an error attempting to connect using Kerberos authentication:

Symptom	Common solutions
"Kerberos logins are not supported" error and the LogFile includes the message "Failed to load the Kerberos GSS-API library"	• Ensure a Kerberos client is installed on the client computer, including the GSS-API library. • The file specified by LogFile lists the name of the library that it is attempting to load. Verify that the library name is correct, and use the Kerberos connection parameter to specify the correct library name, if necessary. • Ensure that the directory including any supporting libraries is listed in the library path (%PATH% on Windows). • If the LogFile output states the GSS-API library was missing entry points, then the library is not a supported Kerberos Version 5 GSS-API library.
"Kerberos logins are not supported" error	• Ensure the database server has enabled Kerberos logins by specifying one or more of the -krb, -kl, or -kr server options. • Ensure Kerberos logins are supported on both the client and server platforms.

Symptom	Common solutions
"Kerberos login failed" error	• Ensure the user is logged into Kerberos and has a valid ticket-granting ticket that has not expired. • Ensure the client computer and server computer both have their time synchronized to within less than 5 minutes.
"Login mode 'Kerberos' not permitted by login_mode setting" error	• The public or temporary public database option setting for the login_mode option must include the value Kerberos to allow Kerberos logins.
"The login ID '`<client-Kerberos-principal>`' has not been mapped to any database user ID"	• The Kerberos principal must be mapped to a database user ID using the GRANT KERBEROS LOGIN statement. Note that the full client principal including the realm must be provided to the GRANT KERBEROS LOGIN statement, and principals which differ only in the instance or realm are treated as different. • Alternatively, if you want any valid Kerberos principal which has not be explicitly mapped to be able to connect, create the guest database user ID with a password using GRANT CONNECT.

Related Information

-z Database Server Option
Kerberos Clients [page 220]

11.2.8 Security: Use Login Modes to Secure the Database

There are several measures you can take to secure your database.

As database options apply only to the database in which they are found, different databases can have a different login settings even if they are loaded and running on the same server.

The login_mode database option accepts the following values:

Standard

Standard user authentication is permitted. This is the default setting. A database user ID and password must be provided.

Integrated

Windows integrated logins are permitted.

Kerberos

Kerberos logins are permitted.

LDAPUA

LDAP (Lightweight Directory Access Protocol) user authenticated logins are permitted.

PAMUA

PAM (Pluggable Authentication Modules) user authenticated logins are permitted.

> ⚠ Caution
>
> Setting the login_mode database option to not allow Standard user authentication (user ID and password) restricts connections to only those users or groups who have been granted an Integrated, Kerberos, LDAPUA, PAMUA, or CloudAdmin login mapping. Attempting to connect with a user ID and password generates an error unless you are a user with the MANAGE ANY USER privilege.

Both login_mode and login_policy options have no influence on the order in which authentication is executed. The order is roughly defined as follows:

- Integrated
- Kerberos
- LDAPUA
- PAMUA
- Standard

Setting the value of the login_mode option for a given database to allow a combination of Standard, Integrated, Kerberos, LDAPUA, and PAMUA logins by using the SET OPTION statement permanently enables the specified types of logins for that database. When you enable Integrated, Kerberos, LDAPUA, or PAMUA logins for your database, you rely on the security model of the operating system or network. For example, the following statement permanently enables Standard and Integrated logins:

```
SET OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

If the database is shut down and restarted, the option value remains the same and Integrated logins remain enabled.

Setting the login_mode option using SET TEMPORARY OPTION still allows user access via Integrated logins, but only until the database is shut down. The following statement changes the option value temporarily:

```
SET TEMPORARY OPTION PUBLIC.login_mode = 'Standard,Integrated';
```

If the permanent option value is Standard, the database will revert to that value when it is shut down.

Setting temporary public options can provide additional security for your database. If the database file is copied to another computer, then Integrated, Kerberos, LDAPUA, and PAMUA logins will not be enabled by default.

If a database contains sensitive information, the computer where the database files are stored should be protected from unauthorized access. Otherwise, the database files could be copied and unauthorized access to the data could be obtained on another computer.

To increase database security:

- Make passwords complex and difficult to guess.
- Strongly encrypt the database file using the AES encryption features of SAP IQ. The encryption key should be complex and difficult to guess.

- Set the permanent PUBLIC.login_mode database option to Standard. To enable Integrated or Kerberos logins, only the temporary public option should be changed each time the server is started. This ensures that only Standard logins are allowed if the database is copied.

11.3 PAM User Authentication

Pluggable Authentication Module (PAM) support allows you to write programs that rely on authentication independently from the underlying authentication scheme.

PAM authentication centralizes user authentication in a separate external system-wide module. If you already use PAM for other databases, such as Adaptive Server Enterprise, you can use the same credentials on all of your systems.

You can "plug in" any authentication mechanism into a PAM system by writing an authentication module. To configure PAM to use that module, add the module to the service name, the set of rules used in PAM authentication.

PAM User Authentication (PAMUA) is available on all supported UNIX and Linux platforms. To configure PAM for your platform, see your operating system documentation.

You must upgrade your database to a version that supports PAMUA, or create a new database, to use PAMUA.

In this section:

11.3.1 Enabling PAM User Authentication

Enable PAM user authentication in the database.

Prerequisites

- PAM is configured on the UNIX system. (See your operating system documentation.)
- Step 2 requires the SET ANY SECURITY OPTION system privilege.
- Steps 3 and 4 require the MANAGE ANY LOGIN POLICY system privilege.

Procedure

1. Configure PAM on your Linux or UNIX system. For this example, assume a PAM service name **PAM_Rule**.
2. Add the PAM value to the LOGIN_MODE database option:

   ```
   SET OPTION PUBLIC.login_mode = PAMUA
   ```

3. Create a login policy that assigns values to pam_service_name and (optionally) pam_failover_to_std:

   ```
   CREATE LOGIN POLICY pam_policy
   ```

   ```
   pam_service_name = PAM_Rule
   ```

   ```
   pam_failover_to_std = ON
   ```

4. Assign the login policy to the users of PAMUA:

   ```
   ALTER USER pam_userID LOGIN POLICY pam_policy
   ```

   You can also create new users and assign the login policy pam_policy to them with a `CREATE USER` statement.
5. Verify that users can log on using PAM user authentication. Use the `dbping` utility to test the database connection:

   ```
   dbping –c “uid=pam_userID;pwd=<pam_user_password>;
   links=tcpip(host=iq_server;port=6263);eng=IQdb” –d
   ```

11.3.2 Sample PAM Authorization Program

This sample program accepts credentials as command line parameters and uses PAM to authenticate them.

The program demonstrates basic authentication. It offers guidance for setting up and troubleshooting PAM authentication without the SAP IQ server.

This program was tested on RedHat Linux 6 using GCC 4.2; other UNIX platforms may require changes. Consult your system's PAM application programming interface for help compiling and running PAM client programs.

```
//*****************************
//Sample Application
//
//To compile this program, use:
//
// gcc pam_test.c
//
//*****************************
#include <security/pam_appl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int null_conv(int num_msg, const struct pam_message **msg, struct pam_response
**resp,
void *appdata_ptr)
{
```

```
   *resp=(struct pam_response*)appdata_ptr;
   return PAM_SUCCESS;
}
int authenticate(char *service, char *user, char *pass)
/*************************************************/
{
   pam_handle_t *pamh=NULL;
   struct pam_response *replay=(struct pam_response *)malloc( sizeoff(struct
pam_response) );
   struct pam_conv conv={nul_cov, (void*)reply };
   int retval=pam_start( service, user, &conv, &pamh );
   if( retval==PAM_SUCCESS){
        reply[0].resp=pass;
        reply[0].rep_retcode=0;
        retval=pam_authenticate( pamh, 0);
        pam_end( pamh, PAM_SUCCESS);
   }
   return (retval==PAM_SUCCESS?0:1);
}
int main(int argc, char *argv )
/*****************************/
{
   int retval;
   char *user, *pass, *service;
//
*****************************************************************************
//Accept command line parameters for username, password, and optional
servicename.
//
*****************************************************************************
   if( argc<3||argc >4){
        fprintf(stderr, "Usage: login <username> <password> [<servicename> ]\n");
        exit(1);
   }
   user=argv[1];
   pass=strdup( argv[2]);
   service=(argc>=4) ? argv[3]:"system-auth";
   retval=authenticate( service, user, pass );
   if (retval==PAM_SUCCESS){
         fprintf(stdout, "Authenticated\n");)
   } else {
         fprintf(stdout, "Not Authenticated\n");
   }
   return retval;
}
```

11.3.3 Sample PAM Configuration

Configure PAM to delegate user authentication to the system authentication module, which is also used by other operation system services like `rlogin`, `chsh`, and `gdm`.

This example reuses the common `rlogin` PAM module to allow for system level authentication. The UNIX user ID used for the `rlogin` tool can also be used by the database server for UNIX account authentication. SAP recommends that you use a copy of the `rlogin`.

1. Verify that PAM is installed correctly on the UNIX machine using the `pamclient` command, if that utility is available. If `pamclient` is not installed, proceed with the next step.
2. Connect to the database as owner, and enter:

```
SET TEMPORARY option PUBLIC.LOGIN_MODE = 'PAMUA,STANDARD'
CREATE LOGIN POLICY usepam
```

```
PAM_SERVICE_NAME = rlogin
PAM_FAILOVER_TO_STD = ON
```

PAM_SERVICE_NAME is set to the name of the configured PAM service. Please refer to the PAM configuration guide to set up and configure a PAM service. On Linux, PAM service configuration files are in `/etc/pam.d`.

3. Create a user with the login policy:

```
CREATE USER Company-uid LOGIN POLICY usepam
```

4. Log in using the `Company-uid` user ID and password.
5. When finished, drop the user and login policy:

```
DROP USER Company-uid
DROP LOGIN policy usepam
```

12 Advanced Security Options in SAP IQ

The SAP IQ Advanced Security Option supports column encryption, Federal Information Processing Standards (FIPS)-approved network encryption technology, and LDAP and Kerberos authentication for database connections, operating system logins, and network logins. The Advanced Security Option is a separately licensed SAP IQ option.

In this section:

12.1 Column Encryption in SAP IQ

SAP IQ supports user-encrypted columns.

In addition to database encryption at rest, SAP IQ supports user-encrypted columns with the AES_ENCRYPT and AES_DECRYPT functions and the LOAD TABLE ENCRYPTED clause. These functions permit explicit encryption and decryption of column data via calls from the application. Encryption and decryption key management is the responsibility of the application.

In this section:

Related Information

AES_ENCRYPT Function [String]
AES_DECRYPT Function [String]
Database Options for Column Encryption [page 240]
LOAD TABLE ENCRYPTED Clause [page 238]

12.1.1 Licensing Requirements for Column Encryption

The Advanced Security Option (IQ_SECURITY) is required to use user-encrypted columns with SAP IQ.

12.1.2 Definitions of Encryption Terms

Definitions of terms used when describing encryption of stored data.

- AES – the Advanced Encryption Standard, a FIPS-approved cryptographic algorithm for the protection of sensitive (but unclassified) electronic data. AES adopted the Rijndael algorithm with restrictions on the block sizes and key lengths. AES is the algorithm supported by SAP IQ.
- ARIA – a variant of AES. ARIA is a general purpose block cipher algorithm.
- CBC – cipher block chaining mode encryption.
- ciphertext – data in an unintelligible form that preserves the information content of the plaintext form.
- CTR – counter-based block cipher encryption.
- decryption – the reverse transformation of ciphertext back to plaintext. Also known as deciphering.
- encryption – a reversible transformation of data from plaintext to ciphertext. Also known as enciphering.
- key – a number used to encrypt or decrypt data. Symmetric-key encryption systems use the same key for both encryption and decryption. Asymmetric-key systems use one key for encryption and a different (but mathematically related) key for decryption. The SAP IQ interfaces accept character strings as keys.
- plaintext – data in its original, intelligible form. Plaintext is not limited to string data, but is used to describe any data in its original representation.
- Rijndael – pronounced “reign dahl.” A specific encryption algorithm that supports a variety of key and block sizes. The algorithm was designed to use simple whole-byte operations and thus is relatively easy to implement in software.

12.1.3 Data Types for Encrypted Columns

The data types supported for encrypted columns and working with these data types.

In this section:

Supported Data Types [page 236]
The first parameter of the AES_ENCRYPT function must be one of the supported data types.

Preservation of Data Types [page 237]
SAP IQ ensures that the original data type of plaintext is preserved when decrypting data, if the AES_DECRYPT function is given the data type as a parameter, or is within a CAST function.

Effect of Different Data Types on Ciphertext [page 237]
To produce identical ciphertext for different datatypes, cast the input of AES_ENCRYPT to the same data type to produce identical ciphertext.

Related Information

AES_ENCRYPT Function [String]
AES_DECRYPT Function [String]
LOAD TABLE ENCRYPTED Clause [page 238]

12.1.3.1 Supported Data Types

The first parameter of the AES_ENCRYPT function must be one of the supported data types.

CHAR	NUMERIC
VARCHAR	FLOAT
TINYINT	REAL
SMALLINT	DOUBLE
INTEGER	DECIMAL
BIGINT	DATE
BIT	TIME
BINARY	DATETIME
VARBINARY	TIMESTAMP
UNSIGNED INT	SMALLDATETIME
UNSIGNED BIGINT	

The LOB data type is not currently supported for SAP IQ column encryption.

12.1.3.2 Preservation of Data Types

SAP IQ ensures that the original data type of plaintext is preserved when decrypting data, if the AES_DECRYPT function is given the data type as a parameter, or is within a CAST function.

SAP IQ compares the target data type of the CAST function with the data type of the originally encrypted data. If the two data types do not match, you see a -1001064 error that includes details about the original and target data types.

For example, given an encrypted VARCHAR(1) value and this valid decryption statement:

```
SELECT AES_DECRYPT ( thecolumn, 'theKey',
VARCHAR(1) ) FROM thetable
```

If you attempt to decrypt the data using:

```
SELECT AES_DECRYPT ( thecolumn, 'theKey',
SMALLINT ) FROM thetable
```

the error returned is:

```
Decryption error: Incorrect CAST type smallint(5,0)
for decrypt data of type varchar(1,0).
```

This data type check is made only when the CAST or the data type parameter are supplied. Otherwise, the query returns the ciphertext as binary data.

When using the AES_ENCRYPT function on literal constants, as in this statement:

```
INSERT INTO t (cipherCol) VALUES (AES_ENCRYPT (1, 'key'))
```

the data type of 1 is ambiguous; it can be a TINYINT, SMALLINT, INTEGER, UNSIGNED INT, BIGINT, UNSIGNED BIGINT, or possibly other data types.

You should explicitly use the CAST function to resolve any potential ambiguity, as in:

```
INSERT INTO t (cipherCol)
VALUES ( AES_ENCRYPT (CAST (1 AS UNSIGNED INTEGER), 'key'))
```

Explicitly converting the data type using the CAST function when encrypting data prevents problems using the CAST function when the data is decrypted.

There is no ambiguity if the data being encrypted is from a column, or if the encrypted data was inserted by LOAD TABLE.

12.1.3.3 Effect of Different Data Types on Ciphertext

To produce identical ciphertext for different datatypes, cast the input of AES_ENCRYPT to the same data type to produce identical ciphertext.

The ciphertext produced by AES_ENCRYPT differs for two different data types given the same input value and same key. A join of two ciphertext columns that holds encrypted values of two different data types may therefore not return identical results.

For example, assume:

```
CREATE TABLE tablea(c1 int, c2 smallint);
INSERT INTO tablea VALUES (100,100);
```

The value `AES_ENCRYPT(c1, 'key')` differs from `AES_ENCRYPT(c2,'key')` and the value `AES_ENCRYPT(c1,'key')` differs from `AES_ENCRYPT(100,'key')`.

To resolve this issue, cast the input of `AES_ENCRYPT` to the same data type. For example, the results of these code fragments are the same:

```
AES_ENCRYPT(c1, 'key');
```

```
AES_ENCRYPT(CAST(c2 AS INT), 'key');
```

```
AES_ENCRYPT(CAST(100 AS INT), 'key');
```

Related Information

AES_ENCRYPT Function [String]

12.1.4 LOAD TABLE ENCRYPTED Clause

The LOAD TABLE statement supports the column-spec keyword ENCRYPTED.

The `<column-specs>` must follow the column name in a LOAD TABLE statement in this order:

- `<format-specs>`
- `<null-specs>`
- `<encrypted-specs>`

Syntax

```
| ENCRYPTED(<data-type> '<key-string>' [, '<algorithm-string>' ] )
```

Parameters

data-type the data type that the input file field should function. `<data-type>` should be the same as the data type of the output of the `AES_DECRYPT` function.

key-string the encryption key used to encrypt the data. This key must be a string literal. To obtain the original value, use the same key to decrypt the value. This parameter is case-sensitive, even in case-insensitive databases.

As you should for most passwords, choose a key value that cannot be easily guessed. Choose a value for that is at least 16 characters long, contains a mix of uppercase and lowercase letters, and includes numbers and special characters. You will need this key each time you want to decrypt the data.

> ⚠ Caution
>
> Protect your key; store a copy of your key in a safe location. A lost key results in the encrypted data becoming completely inaccessible, from which there is no recovery.

algorithm-string the algorithm used to encrypt the data. This parameter is optional, but data must be encrypted and decrypted using the same algorithm. Currently, AES128 is the default, as it is the only supported algorithm. AES128 is a block encryption algorithm chosen as the new Advanced Encryption Standard (AES) for block ciphers by the National Institute of Standards and Technology (NIST).

Usage

The ENCRYPTED column specification allows you to specify the encryption key and, optionally, the algorithm to use to encrypt the data that is loaded into the column. The target column for this load should be the data type that the input file field should be converted to as input to the `VARBINARY`. Specifying other data types returns an error.

Example

```
LOAD TABLE <table_name>
(
<plaintext_column_name>,
<a_ciphertext_column_name>
NULL('nil')
ENCRYPTED(varchar(6),'tHefiRstkEy') ,
<another_encrypted_column>
ENCRYPTED(bigint,'thEseconDkeY','AES')
)
FROM '/path/to/the/input/file'
FORMAT ascii
DELIMITED BY ';'
ROW DELIMITED BY '\0xa'
QUOTES OFF
ESCAPES OFF
```

where the format of the input file for the `LOAD TABLE` statement is:

```
a;b;c;
d;e;f;
g;h;i;
```

Related Information

LOAD TABLE Statement
Data Types for Encrypted Columns [page 235]

12.1.5 String Comparisons on Encrypted Text

If data is case-insensitive, or uses a collation other than ISO_BINENG, you must decrypt ciphertext columns to perform string comparisons.

When performing comparisons on strings, the distinction between equal and identical strings is important for many collations and depends on the CASE option of CREATE DATABASE. In a database that is set to CASE RESPECT and uses the ISO_BINENG collation, the defaults for SAP IQ, equality, and identity questions are resolved the same way.

Identical strings are always equal, but equal strings may not be identical. Strings are identical only if they are represented using the same byte values. When data is case-insensitive or uses a collation where multiple characters must be treated as equal, the distinction between equality and identity is significant. ISO1LATIN1 is such a collation.

For example, the strings "ABC" and "abc" in a case-insensitive database are not identical but are equal. In a case-sensitive database, they are neither identical nor equal.

The ciphertext created by the SAP encryption functions preserves identity but not equality. In other words, the ciphertext for "ABC" and "abc" will never be equal.

To perform equality comparisons on ciphertext when your collation or CASE setting does not allow this type of comparison, your application must modify the values within that column into some canonical form, where there are no equal values that are not also identical values. For example, if your database is created with CASE IGNORE and the ISO_BINENG collation and your application applies UCASE to all input values before placing them into the column, then all equal values are also identical.

12.1.6 Database Options for Column Encryption

Certain SAP IQ database option settings affect column encryption and decryption; the default settings are not optimal for most column encryption operations.

In this section:

Protect Ciphertext from Accidental Truncation [page 241]
To prevent accidental truncation of the ciphertext output of the encrypt function (or accidental truncation of any other character or binary string), set the STRING_RTRUNCATION database option.

Preserve Ciphertext Integrity [page 241]
Set ASE_BINARY_DISPLAY to preserve ciphertext integrity.

Prevent Misuse of Ciphertext [page 242]
Set CONVERSION_MODE to prevent implicit data type conversions of encrypted data that result in semantically meaningless operations.

12.1.6.1 Protect Ciphertext from Accidental Truncation

To prevent accidental truncation of the ciphertext output of the encrypt function (or accidental truncation of any other character or binary string), set the STRING_RTRUNCATION database option.

```
SET OPTION STRING_RTRUNCATION = 'ON'
```

When `STRING_RTRUNCATION` is ON (the default), the engine raises an error whenever a string would be truncated during a load, insert, update, or `SELECT INTO` operation. This is ISO/ANSI SQL behavior and is a recommended practice.

When explicit truncation is required, use a string expression such as `LEFT`, `SUBSTRING`, or `CAST`.

Setting `STRING_RTRUNCATION` OFF forces silent truncation of strings.

The `AES_DECRYPT` function also checks input ciphertext for valid data length, and checks text output to verify both the resulting data length and the correctness of the supplied key. If you supply the data type argument, the data type is checked as well.

Related Information

STRING_RTRUNCATION Option [TSQL]
AES_ENCRYPT Function [String]
AES_DECRYPT Function [String]
REPEAT Function [String]

12.1.6.2 Preserve Ciphertext Integrity

Set `ASE_BINARY_DISPLAY` to preserve ciphertext integrity.

```
SET OPTION ASE_BINARY_DISPLAY = 'OFF'
```

When `ASE_BINARY_DISPLAY` is OFF (the default), the system leaves binary data unmodified, and in its raw binary form.

When `ASE_BINARY_DISPLAY` is ON, the system converts binary data into its hexadecimal string display representation. Temporarily set the option ON only if you need to show data to an end user, or if you need to export the data to another external system, where raw binary may become altered in transit.

12.1.6.3 Prevent Misuse of Ciphertext

Set `CONVERSION_MODE` to prevent implicit data type conversions of encrypted data that result in semantically meaningless operations.

The `CONVERSION_MODE` database option restricts implicit conversion between binary data types (`BINARY`, `VARBINARY`, and `LONG BINARY`) and other nonbinary data types (`BIT`, `TINYINT`, `SMALLINT`, `INT`, `UNSIGNED INT`, `BIGINT`, `UNSIGNED BIGINT`, `CHAR`, `VARCHAR`, and `LONG VARCHAR`) on various operations:

```
SET TEMPORARY OPTION CONVERSION_MODE = 1
```

Setting `CONVERSION_MODE` to 1 restricts implicit conversion of binary data types to any other nonbinary data type on `INSERT` and `UPDATE` commands, and in queries. The restrict binary conversion mode also applies to `LOAD TABLE` default values and `CHECK` constraint.

The `CONVERSION_MODE` option default value of 0 maintains the implicit conversion behavior of binary data types in versions of SAP IQ earlier than 12.7.

Related Information

CONVERSION_MODE Option

12.1.7 Encryption and Decryption Example

An example using the `AES_ENCRYPT` and `AES_DECRYPT` functions, written in commented SQL.

```
--   This example of aes_encrypt and aes_decrypt function use is presented in
three parts:
--
-- Part I: Preliminary description of target tables and users as DDL
-- Part II: Example schema changes motivated by introduction of encryption
-- Part III: Use of views and stored procedures to protect encryption keys
--
--   Part I: Define target tables and users
--   Assume two classes of user, represented here by the instances
--   PrivUser and NonPrivUser, assigned to groups reflecting differing
--   privileges.
--   The initial state reflects the schema prior to the introduction
--   of encryption.
--   Set up the starting context: There are two tables with a common key.
--   Some columns contain sensitive data, the remaining columns do not.
--   The usual join column for these tables is sensitiveA.
--   There is a key and a unique index.
    grant connect to PrivUser identified by 'verytrusted' ;
    grant connect to NonPrivUser identified by 'lesstrusted' ;
    grant connect to high_privileges_group ;
    create role high_privileges_group ;
    grant role high_privileges_group to PrivUser ;
    grant connect to low_privileges_group ;
    create role low_privileges_group ;
    grant role low_privileges_group to NonPrivUser ;
    create table DBA.first_table
                    (sensitiveA char(16) primary key
```

```
                    ,sensitiveB numeric(10,0)
                    ,publicC    varchar(255)
                    ,publicD    date
                    )  ;
--   There is an implicit unique HG (High_Group) index enforcing the primary key.
    create table second_table
                    (sensitiveA char(16)
                    ,publicP integer
                    ,publicQ tinyint
                    ,publicR varchar(64)
                    )  ;
    create hg index second_A_HG on second_table ( sensitiveA ) ;
--   TRUSTED users can see the sensitive columns.
    grant select ( sensitiveA, sensitiveB, publicC, publicD )
        on DBA.first_table to PrivUser ;
    grant select ( sensitiveA, publicP, publicQ, publicR )
        on DBA.second_table to PrivUser ;
--   Non-TRUSTED users in existing schema need to see sensitiveA to be
--   able to do joins, even though they should not see sensitiveB.
    grant select ( sensitiveA, publicC, publicD )
        on DBA.first_table to NonPrivUser ;
    grant select ( sensitiveA, publicP, publicQ, publicR )
        on DBA.second_table to NonPrivUser ;
--   Non-TRUSTED users can execute queries such as
    select I.publicC, 3*II.publicQ+1
    from DBA.first_table I, DBA.second_table II
    where I.sensitiveA = II.sensitiveA and I.publicD IN ( '2006-01-11' ) ;
--   and
    select count(*)
    from DBA.first_table I, DBA.second_table II
    where I.sensitiveA = II.sensitiveA and SUBSTR(I.sensitiveA,4,3)
    BETWEEN '345' AND '456' ;
--   But only TRUSTED users can execute the query
    select I.sensitiveB, 3*II.publicQ+1
    from DBA.first_table I, DBA.second_table II
    where I.sensitiveA = II.sensitiveA and I.publicD IN ( '2006-01-11' ) ;
--   Part II: Change the schema in preparation for encryption
--
--    The DBA introduces encryption as follows:
--
--    For applicable tables, the DBA changes the schema, adjusts access
--    permissions, and updates existing data. The encryption
--    keys used are hidden in a subsequent step.
--   DataLength comparison for length of varbinary encryption result
--   (units are Bytes):
--
--    PlainText CipherText    Corresponding Numeric Precisions
--
--            0       16
--      1 -  16       32      numeric(1,0)   - numeric(20,0)
--     17 -  32       48      numeric(21,0)  - numeric(52,0)
--     33 -  48       64      numeric(53,0)  - numeric(84,0)
--     49 -  64       80      numeric(85,0)  - numeric(116,0)
--     65 -  80       96      numeric(117,0) - numeric(128,0)
--     81 -  96      112
--     97 - 112      128
--    113 - 128      144
--    129 - 144      160
--    145 - 160      176
--    161 - 176      192
--    177 - 192      208
--    193 - 208      224
--    209 - 224      240
--   The integer data types tinyint, small int, integer, and bigint
--   are varbinary(32) ciphertext.
--   The exact relationship is
--   DATALENGTH(ciphertext) =
--   (((DATALENGTH(plaintext)+ 15) / 16) + 1) * 16
```

```
--   For the first table, the DBA chooses to preserve both the plaintext and
--   ciphertext forms. This is not typical and should only be done if the
--   database files are also encrypted.
--   Take away NonPrivUser's access to column sensitiveA and transfer
--   access to the ciphertext version.
--   Put a unique index on the ciphertext column. The ciphertext
--   itself is indexed.
--   NonPrivUser can select the ciphertext and use it.
--   PrivUser can still select either form (without paying decrypt costs).
    revoke select ( sensitiveA ) on DBA.first_table from NonPrivUser ;
    alter table DBA.first_table add encryptedA varbinary(32) ;
    grant select ( encryptedA ) on DBA.first_table to PrivUser ;
    grant select ( encryptedA ) on DBA.first_table to NonPrivUser ;
    create unique hg index first_A_unique on first_table ( encryptedA ) ;
    update DBA.first_table
        set encryptedA = aes_encrypt(sensitiveA, 'seCr3t')
        where encryptedA is null ;
    commit
--   Now change column sensitiveB.
    alter table DBA.first_table add encryptedB varbinary(32) ;
    grant select ( encryptedB ) on DBA.first_table to PrivUser ;
    create unique hg index first_B_unique on first_table ( encryptedB ) ;
    update DBA.first_table
        set encryptedB = aes_encrypt(sensitiveB,
        'givethiskeytonoone') where encryptedB is null ;
    commit
--   For the second table, the DBA chooses to keep only the ciphertext.
--   This is more typical and encrypting the database files is not required.
    revoke select ( sensitiveA ) on DBA.second_table from NonPrivUser ;
    revoke select ( sensitiveA ) on DBA.second_table from PrivUser ;
    alter table DBA.second_table add encryptedA varbinary(32) ;
    grant select ( encryptedA ) on DBA.second_table to PrivUser ;
    grant select ( encryptedA ) on DBA.second_table to NonPrivUser ;
    create unique hg index second_A_unique on second_table ( encryptedA ) ;
    update DBA.second_table
        set encryptedA = aes_encrypt(sensitiveA, 'seCr3t')
        where encryptedA is null ;
    commit
    alter table DBA.second_table drop sensitiveA ;
--   The following types of queries are permitted at this point, before
--   changes are made for key protection:
--   Non-TRUSTED users can equi-join on ciphertext; they can also select
--   the binary, but have no way to interpret it.
    select I.publicC, 3*II.publicQ+1
    from DBA.first_table I, DBA.second_table II
    where I.encryptedA = II.encryptedA and I.publicD IN ( '2006-01-11' ) ;
--   Ciphertext-only access rules out general predicates and expressions.
--   The following query does not return meaningful results.
--
--   NOTE: These four predicates can be used on the varbinary containing
--   ciphertext:
--      = (equality)
--      <> (inequality)
--      IS NULL
--      IS NOT NULL
    select count(*)
    from DBA.first_table I, DBA.second_table II
    where I.encryptedA = II.encryptedA and SUBSTR(I.encryptedA,4,3)
        BETWEEN '345' AND '456' ;
--   The TRUSTED user still has access to the plaintext columns that
--   were retained. Therefore, this user does not need to call
--   aes_decrypt and does not need the key.
    select count(*)
    from DBA.first_table I, DBA.second_table II
    where I.encryptedA = II.encryptedA and SUBSTR(I.sensitiveA,4,3)
        BETWEEN '345' AND '456' ;
--   Part III: Protect the encryption keys
--   This section illustrates how to grant access to the plaintext, but
```

```
--   still protect the keys.
--   For the first table, the DBA elected to retain the plaintext columns.
--   Therefore, the following view has the same capabilities as the trusted
--   user above.
--   Assume group_member is being used for additional access control.
--   NOTE: In this example, NonPrivUser still has access to the ciphertext
--   encrypted in the base table.
    create view DBA.a_first_view (sensitiveA, publicC, publicD)
        as
            select
              IF group_member('high_privileges_group',user_name()) = 1
                 THEN sensitiveA
                 ELSE NULL
              ENDIF,
              publicC,
              publicD
            from first_table ;
    grant select on DBA.a_first_view to PrivUser ;
    grant select on DBA.a_first_view to NonPrivUser ;
--   For the second table, the DBA did not keep the plaintext.
--   Therefore, aes_decrypt calls must be used in the view.
--   IMPORTANT: Hide the view definition with ALTER VIEW, so that no one
--   can discover the key.
    create view DBA.a_second_view (sensitiveA,publicP,publicQ,publicR)
        as
            select
              IF group_member('high_privileges_group',user_name()) = 1
                 THEN aes_decrypt(encryptedA,'seCr3t', char(16))
                 ELSE NULL
              ENDIF,
              publicP,
              publicQ,
              publicR
            from second_table ;
    alter view DBA.a_second_view set hidden ;
    grant select on DBA.a_second_view to PrivUser ;
    grant select on DBA.a_second_view to NonPrivUser ;
--   Likewise, the key used for loading can be protected in a stored
procedure.
--   By hiding the procedure (just as the view is hidden), no-one can see
--   the keys.
    create procedure load_first_proc(@inputFileName varchar(255),
                        @colDelim varchar(4) default '$',
                        @rowDelim varchar(4) default '\n')
        begin
            execute immediate with quotes
                'load table DBA.second_table
                (encryptedA encrypted(char(16),' ||
                '''' || 'seCr3t' || '''' || '),publicP,publicQ,publicR) ' ||
                ' from ' || '''' || @inputFileName || '''' ||
                ' delimited by ' || '''' || @colDelim || '''' ||
                ' row delimited by ' || '''' || @rowDelim || '''' ||
                ' quotes off escapes off' ;
        end
    ;
    alter procedure DBA.load_first_proc set hidden ;
--   Call the load procedure using the following syntax:
    call load_first_proc('/dev/null', '$', '\n') ;
--   Below is a comparison of several techniques for protecting the
--   encryption keys by using user-defined functions (UDFs), other views,
--   or both. The first and the last alternatives offer maximum performance.
--   The second_table is secured as defined earlier.
--   Alternative 1:
--   This baseline approach relies on restricting access to the entire view.
     create view
            DBA.second_baseline_view(sensitiveA,publicP,publicQ,publicR)
        as
            select
```

```
              IF group_member('high_privileges_group',user_name()) = 1
                 THEN aes_decrypt(encryptedA,'seCr3t', char(16))
                 ELSE NULL
              ENDIF,
              publicP,
              publicQ,
              publicR
            from DBA.second_table ;
     alter view DBA.second_baseline_view set hidden ;
     grant select on DBA.second_baseline_view to NonPrivUser ;
     grant select on DBA.second_baseline_view to PrivUser ;
--   Alternative 2:
--     Place the encryption function invocation within a user-defined
--   function (UDF).
--    Hide the definition of the UDF. Restrict the UDF permissions.
--    Use the UDF in a view that handles the remainder of the security
--     and business logic.
--    Note: The view itself does not need to be hidden.
     create function DBA.second_decrypt_function(IN datum varbinary(32))
            RETURNS char(16) DETERMINISTIC
            BEGIN
                RETURN aes_decrypt(datum,'seCr3t', char(16));
            END  ;
     grant execute on DBA.second_decrypt_function to PrivUser ;
     alter function DBA.second_decrypt_function set hidden ;
     create view
            DBA.second_decrypt_view(sensitiveA,publicP,publicQ,publicR)
        as
                select
                  IF group_member('high_privileges_group',user_name()) = 1
                     THEN second_decrypt_function(encryptedA)
                     ELSE NULL
                  ENDIF,
                  publicP,
                 publicQ,
                 publicR
                from DBA.second_table ;
     grant select on DBA.second_decrypt_view to NonPrivUser ;
     grant select on DBA.second_decrypt_view to PrivUser ;
--  Alternative 3:
--    Sequester only the key selection in a user-defined function.
--    This function could be extended to support selection of any
--  number of keys.
--    This UDF is also hidden and has restricted execute privileges.
--    Note: Any view that uses this UDF therefore does not compromise
-- the key values.
     create function DBA.second_key_function()
              RETURNS varchar(32) DETERMINISTIC
              BEGIN
                return 'seCr3t' ;
              END
     grant execute on DBA.second_key_function to PrivUser ;
     alter function DBA.second_key_function set hidden ;
     create view DBA.second_key_view(sensitiveA,publicP,publicQ,publicR)
                as
                    select
                      IF group_member('high_privileges_group',user_name()) = 1
                         THEN aes_decrypt(encryptedA,second_key_function(),
                         char(16))
                         ELSE NULL
                      ENDIF,
                      publicP,
                      publicQ,
                      publicR
                    from DBA.second_table ;
     grant select on DBA.second_key_view to NonPrivUser ;
     grant select on DBA.second_key_view to PrivUser ;
--  Alternative 4:
```

```
--    The recommended alternative is to separate the security logic
--    from the business logic by dividing the concerns into two views.
--    Only the security logic view needs to be hidden.
--    Note: The performance of this approach is similar to that of the first
-- alternative.
     create view
        DBA.second_SecurityLogic_view(sensitiveA,publicP,publicQ,publicR)
            as
                select
                  IF group_member('high_privileges_group',user_name()) = 1
                     THEN aes_decrypt(encryptedA,'seCr3t', char(16))
                     ELSE NULL
                  ENDIF,
                  publicP,
                  publicQ,
                  publicR
                from DBA.second_table ;
     alter view DBA.second_SecurityLogic_view set hidden ;
     create view
        DBA.second_BusinessLogic_view(sensitiveA,publicP,publicQ,publicR)
            as
                select
                  sensitiveA,
                  publicP,
                  publicQ,
                  publicR
                from DBA.second_SecurityLogic_view ;
     grant select on DBA.second_BusinessLogic_view to NonPrivUser ;
     grant select on DBA.second_BusinessLogic_view to PrivUser ;
-- End of encryption example
```

Related Information

AES_DECRYPT Function [String]
AES_ENCRYPT Function [String]
LOAD TABLE ENCRYPTED Clause [page 238]

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering an SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap