Security Management ............................................................1
  Plan and Implement Role-Based Security ......................2
  Roles ...............................................................................2
    User-Defined Roles ..........................................................3
    System Roles .....................................................................22
    Compatibility Roles ..........................................................28
  Views, Procedures and Tables That Are Owned
    by Roles ........................................................................37
  Display Roles Granted ..........................................................38
  Determine Roles and Privileges Granted to a
    User ..................................................................................39
  Privileges ...........................................................................40
    Privileges Versus Permissions ...............................................40
    System Privileges ................................................................41
    Object-Level Privileges .......................................................84
    System Procedure Privileges ...............................................96
  Manage Passwords ................................................................101
    Granting the CHANGE PASSWORD System
      Privilege to a User .........................................................101
    Revoking the CHANGE PASSWORD System
      Privilege from a User .......................................................104
    Changing a Password – Single Control .........................105
    Dual Control Password Management Option ..................106
    Changing a Password – Dual Control ..............................107
  Impersonation .....................................................................108
    Understand the Requirements for
      Impersonation ................................................................110
    Granting the SET USER System Privilege to a
      User ................................................................................113
    Start Impersonating Another User ....................................115
Verify the Current Impersonation Status of a User ............................................................... 117
Stop Impersonating Another User ..................... 117
Revoking the SET USER System Privilege from a User ............................................................ 118
Users .......................................................................... 119
DBA User ........................................................... 120
Super-User ........................................................ 121
Increase Password Security ..................... 122
Case-sensitivity of User IDs and Passwords ...... 122
Creating a New User ......................................... 123
Deleting a User .................................................. 123
Changing a User's Password ..................... 124
Converting a User-Extended Role Back to a User ........................................................................ 125
Permanently Locking a User Account ........ 125
Unlocking User Accounts .............................. 126
Automatic Unlocking of User Accounts ......... 127
Login Policies .............................................................. 128
Modifying the Root Login Policy .................. 128
Creating a New Login Policy ..................... 129
Modifying an Existing Login Policy ............. 129
Deleting a Login Policy ............................... 130
Assigning a Login Policy When Creating a New User ........................................................... 130
Assigning a Login Policy to an Existing User .... 131
User Connections ....................................................... 131
Preventing Connection After Failed Login Attempts ............................................................... 132
Creating a DBA Recovery Account ............. 133
Logging in with a DBA Recovery Account ...... 133
Manage Connections Using Stored Procedures ............................................................... 133
Manage Resources Used by Connections ....... 134
Security with Views and Procedures ............. 136
Security Management

SAP® Sybase® IQ provides a role-based security model to control access to database objects and the execution of privileged operations. A role-based security model provides complete control and granularity for the privileges you want to grant to users. Each privileged operation a user can perform in the database requires one or more system privilege or object-level privilege.

A system privilege is a right to perform an authorized database task. For example, the CREATE TABLE system privilege allows a user to create self-owned tables.

An object-level privilege is a right to perform an authorized task on a specified object. For example, having ALTER privileges on TableA allows a user to alter that table, but not other tables.

A role is a container which 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 give users 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 the user.
- Assign a login policy to the user. By default, a user is assigned to the root login policy.
- Set the user as the publisher or as a remote user of the database for use in a SQL Remote system.

Each new or migrated SAP Sybase 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 are a pre-16.0 SAP Sybase IQ customer, it is recommended that you review the sections on how the security model has changed from the authority/permission/group model to the role/privilege/user-extended role model under *Upgrading to Role-Based Security* in the Migration document appropriate to your operating system.
Plan and Implement Role-Based Security

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

Design 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. A role hierarchy which matches the organizational hierarchy can be created. Assign a name to each grouping. These groupings are the roles you will 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 administrators for the roles you are going to 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 will be creating.

Build the Security Hierarchy

1. Create the required roles. See Roles on page 2.
2. To each role, grant the system privileges. See Roles on page 2 and Privileges on page 40.
3. Create the users. See Users on page 119.
4. Grant the applicable roles to each user, including granting administrative rights where applicable. See Roles on page 2
5. Grant the applicable object-level and system privileges to users, including granting administrative rights where applicable. See Privileges on page 40.

See also
- Roles on page 2
- Privileges on page 40
- Users on page 119

Roles

A role is a container which may contain system privileges, object-level privileges, and roles. Granting privileges to and revoking privileges from a role is the same as for a user. A role and user cannot have the same name.

There are three types of roles:
• **User-Defined Role** – a custom collection of system and object-level privileges, typically created to group privileges related to a specific task or set of tasks. You can create user-defined roles to suit the needs of your organization.

• **System Role** – a built-in role automatically created in each newly created database. A system role can be granted and revoked, but cannot be dropped. With some exceptions, their default underlying system privileges cannot be modified or revoked, but additional roles and system privileges can be granted to and revoked from a system role.

• **Compatibility Role** – created for backwards compatibility with earlier versions of SAP Sybase IQ, you can grant, revoke, and under specific conditions, drop them. You cannot, however, modify their underlying system privileges.

### User-Defined Roles

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

A user-defined role:

• Can be a standalone object with no login privileges, which can own objects.
• Can be a database user with the ability to act as a role (user-extended role). If the original user has login privileges, the user-extended role inherits the login privileges.
• Can be granted privileges on other objects.
• Can be granted privileges of other roles.
• Has a case-insensitive name.

The granting of a user-defined role is semantically equivalent to individually granting each of its underlying system and object-level privileges.

A user-defined role can be granted with or without administrative rights. When granted with administrative rights, a user can manage (grant, revoke, and drop) the role, as well as use any of the underlying system and object-level privileges of the role. When granted with administrative rights only, a user can manage the role, but cannot use its underlying system and object-level privileges. When granted with no administrative rights, a user can use its underlying system and object-level privileges, but cannot manage the role.

Extending a user to act as a role is useful when you have a user with a set of system and object-level privileges that you want to grant to another user.

You cannot convert a user-defined role to a user-extended role, and vice versa.

When you grant a user-extended role to a user or another role, the grantee inherits all the system and object-level privileges that the user-extended role has, including any administration rights.

---

**Note:** Unless otherwise noted, the term *user-defined role* refers to both user-extended and user-defined roles.
**Creating a User-Defined Role**
Create a new user-defined role.

**Prerequisites**
Requires the MANAGE ROLES system privilege.

**Task**
A user-defined role cannot have a login password. When creating a user-defined role, you can appoint administrators for the role, and indicate whether they are also to be members of the role. If you do not specify any administrators, the global role administrator (any user granted the MANAGE ROLES system privilege) becomes the default administrator of the role.

However, if at least one role administrator is specified during conversion, global role administrators will be unable to manage the role because the SYS_MANAGE_ROLES_ROLE system privilege is not automatically granted to the role with administrative rights. For this reason, it is strongly recommended that you either do not define any role administrators when creating a role (add them after creation), or explicitly grant the SYS_MANAGE_ROLES_ROLE system privilege with administrative rights only along with any role administrators during the creation process.

Role administrators can be added and removed after creation. When creating a role, if the new role name already exists, the statement fails.

To create a new user-defined role, execute one of these statements:

<table>
<thead>
<tr>
<th>Create Condition</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global role administrator only; no role administrators</td>
<td>CREATE ROLE role_name</td>
</tr>
<tr>
<td>Role administrators with no role membership; no global role administrator</td>
<td>CREATE ROLE role_name WITH ADMIN ONLY admin_name [,...]</td>
</tr>
<tr>
<td>Role administrators with role membership; no global role administrator*</td>
<td>CREATE ROLE role_name WITH ADMIN admin_name [,...]</td>
</tr>
<tr>
<td>Role administrators with no role membership; with global role administrator*</td>
<td>CREATE ROLE role_name WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, admin_name [,...]</td>
</tr>
</tbody>
</table>

*Since global role administrators cannot be granted membership in a role, you cannot include SYS_MANAGE_ROLES_ROLE in the administrators list when creating a role with role
administrators granted membership in the role (WITH ADMIN option). It can, however, be included when creating a role with role administrators not granted membership in the role (WITH ADMIN ONLY option).

Example:

This statement creates the role Sales with no role administrators specified. Any user with the MANAGE ROLES system privilege is a default administrator of this role.

```sql
CREATE ROLE Sales
```

This statement creates the role Marketing with Jane and Bob acting as role administrators, but are not granted membership in the role. It also allows global role administrators to manage the role.

```sql
CREATE ROLE Marketing WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, jane, bob
```

See also
- Role and Global Role Administrators on page 10
- CREATE ROLE Statement on page 254

Converting an Existing User to a User-Extended Role

You can extend an existing user ID to act as a role. If an original user has login privileges, the user-extended role retains the login privileges.

Prerequisites

Requires the MANAGE ROLES system privilege.

Task

When converting a user to act as a role, you can appoint administrators for the role, and indicate whether they are also to be members of the role. If you do not specify any administrators, the global role administrator (any user granted the MANAGE ROLES system privilege) becomes the default administrator of the role.

However, if at least one role administrator is specified during conversion, global role administrators will be unable to manage the role because the SYS_MANAGE_ROLES_ROLE system privilege is not automatically granted to the role with administrative rights. For this reason, it is strongly recommended that you either do not define any role administrators when creating a role (add them after creation), or explicitly grant the SYS_MANAGE_ROLES_ROLE system privilege with administrative rights only along with any role administrators during the creation process.

Role administrators can be added and removed after conversion. When converting a user to act as a role, if the specified user ID does not already exist, the statement fails.

To convert an existing user, execute one of these statements:
Convert Condition | Statement
---|---
Global role administrator only; no role administrators | CREATE ROLE FOR USER userID
Role administrators with no role membership; no global role administrator | CREATE ROLE FOR USER userID WITH ADMIN ONLY admin_name [,...]
Role administrators with role membership; no global role administrator* | CREATE ROLE FOR USER userID WITH ADMIN admin_name [,...]
Role administrators with no role membership; global role administrator* | CREATE ROLE FOR USER userID WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, admin_name [,...]

*Since global role administrators cannot be granted membership in a role, you cannot include SYS_MANAGE_ROLES_ROLE in the administrators list when creating a role with role administrators granted membership in the role (WITH ADMIN option). It can, however, be included when creating a role with role administrators not granted membership in the role (WITH ADMIN ONLY option).

Example:
This statement extends user Sales1 to act as a role. Since no role administrators are specified, any user with the MANAGE ROLES system privilege can administrator the role.

```sql
CREATE ROLE FOR USER Sales1
```

This statement extends the user Marketing1 to act as a role, with Jane and Bob acting as role administrators. It also allows global role administrators to manage the role.

```sql
CREATE ROLE FOR USER Marketing1 WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, jane, bob
```

See also
- Role and Global Role Administrators on page 10
- CREATE ROLE Statement on page 254

Converting a User-Extended Role Back to a User
You can convert a user-extended role back to a regular user.

Prerequisites
Requires administrative rights over the user-extended role being converted.
Task

The user retains any login privileges, system privileges and roles granted to the user-extended role. The user remains as the owner of the objects that were created after the user was extended to act as a role. Any members of the user-extended role are immediately revoked.

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 converting a user-extended role back to a user, all dependent roles of the user-extended role must continue to meet this minimum requirement, or the conversion fails.

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

<table>
<thead>
<tr>
<th>Convert Condition</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role has not been granted any members.</td>
<td>DROP ROLE FROM USER role_name</td>
</tr>
<tr>
<td>Role has been granted members.</td>
<td>DROP ROLE FROM USER role_name WITH REVOKE</td>
</tr>
</tbody>
</table>

Adding a User-Defined Role to a User or Role

Add membership in a user-defined role to a user or role (grantee), with or without administrative rights.

Prerequisites

Requires administrative privilege over the role being granted.

Task

When granted with administrative rights, a user can manage (grant, revoke, and drop) the role, as well as use any of the underlying system privileges of the role. When granted with administrative rights only, a user can manage the role, but not use its underlying system privileges. Finally, when granted with no administrative rights, a user can only use its underlying system privileges. If no administrative clause is specified, the role is granted with no administrative rights.

When a user is granted membership in a role, the user inherits all underlying system privileges and roles of the role, including any object-level permissions on tables, views, and procedures.

When a role is granted to another role, all members of the role being granted (the child role) automatically become members of the receiving role (parent role) and inherit all underlying system privileges and roles of the parent role, including those on tables, views, and procedures. Existing members of the parent role do not become members of the child role or inherit any of its underlying system privileges and roles.

To grant a user-defined role to a grantee, execute one of these statements:
<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Membership in the role along with full administrative rights to the role</td>
<td><code>GRANT ROLE role_name TO grantee [,....] WITH ADMIN OPTION</code></td>
</tr>
<tr>
<td>Administrative rights to the role only</td>
<td><code>GRANT ROLE role_name TO grantee [,....] WITH ADMIN ONLY OPTION</code></td>
</tr>
<tr>
<td>Membership in the role, but with no administrative rights to the role</td>
<td><code>GRANT ROLE role_name TO grantee [,....] WITH NO ADMIN OPTION</code></td>
</tr>
</tbody>
</table>

**Example:**

- There are three users: User1, User2, User3.
- There are four roles: Role1, Role2, Role3, Role4.
- There are two system privileges: Priv1, Priv2.
- Role1 is granted Priv1 and Role3.
- User2 and User3 are members of Role1.
- Role2 is granted Priv2 and Role4.
- User3 is a member of Role2.

You execute the following statement:

```
GRANT ROLE Role1 TO User1 WITH ADMIN OPTION
```

User1 becomes a member of Role1.

As a member of Role1, User1 inherits Priv1 and (indirectly) all system privileges and roles from Role3.

User1 can also administer Role1.

You execute the following statement:

```
GRANT ROLE Role2 TO Role1 WITH ADMIN OPTION
```

Role1 becomes a member of Role2.

As members of Role1, User2, User3, and User1 (from previous grant) inherit the following from Role2: Priv2 and (indirectly) all system privileges and roles of Role4.

As a member of Role2, User3 does not become a member of Role1 and does not inherit any system privileges or roles of Role1.

User1, User2, and User3 can administer Role2.
Removing Members from a User-Defined Role

Remove a user or role as a member of a role. The user or role loses the ability to use any underlying system privileges or roles of a role, along with the ability to administer the role, if granted.

Prerequisites
Requires administrative privilege over the role being managed.

Task
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 removing a member from a user-extended role, if the member is an administrator of the role and their removal would violate the minimum requirement, the removal fails.

To remove membership in a user-defined role from a grantee, execute one of these statements:

<table>
<thead>
<tr>
<th>Revoke Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role membership and all administrative rights to the role</td>
<td><code>REVOKE ROLE role_name FROM grantee [...]</code></td>
</tr>
<tr>
<td>Administrative rights to the role only</td>
<td><code>REVOKE ADMIN OPTION FOR ROLE role_name FROM grantee [...]</code></td>
</tr>
</tbody>
</table>

See also
- `REVOKE ROLE Statement` on page 282

Deleting a User-Defined 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 after the drop. If the minimum value is not maintained, the command fails.

Prerequisites
- Requires administrative privilege over the role being dropped.
- If the role being dropped is a user-defined role, the role does not own any objects.
Security Management

Task

A user-defined role can be deleted as long as all dependent roles retain the minimum required number of administrator users with active passwords after the drop. If the minimum value is not maintained, the delete fails.

If a user-extended role is converted back to a user, the objects owned are not deleted. They remain owned by the converted user.

The type of role being deleted and whether it was granted to users determines the clauses required by the DROP statement.

- **FROM USER** – required when deleting a user-extended role.
- **WITH REVOKE** – required to delete a role that has been granted to multiple users and roles.

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

<table>
<thead>
<tr>
<th>Delete Condition</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-defined role has not been granted any members</td>
<td>DROP ROLE role_name</td>
</tr>
<tr>
<td>User-extended role has been granted members</td>
<td>DROP ROLE role_name WITH REVOKE</td>
</tr>
<tr>
<td>User-extended role has not been granted any members*</td>
<td>DROP ROLE FROM USER role_name</td>
</tr>
<tr>
<td>User-extended role has been granted members*</td>
<td>DROP ROLE FROM USER role_name WITH REVOKE</td>
</tr>
</tbody>
</table>

*User-extended role becomes a regular user.

See also

- **DROP ROLE Statement on page 258**

Role and Global Role Administrators

Role administrators and global role administrators are responsible for granting and revoking user-defined roles to users and other roles. You can add and remove role and global role administrators on a role as needed.

There is no maximum number of role administrators that can be granted to a single role. However, there is a minimum number, as specified by the configurable **MIN_ROLE_ADMINS** database option. This minimum requirement is validated before you can revoke a role.
administrator or global role administrator from a role. The minimum number of role administrators can be set to any value between 1 (default) and 10.

A role administrator can be a user, a user-extended role, or a user-defined role.

Global role administrators are any users granted the MANAGE ROLES system privilege. Global role administrators can administer any role to which the SYS_MANAGE_ROLES_ROLE system privilege has been granted with administrative rights.

Both role and global role administrators can grant, revoke, and drop roles, and can add or remove role and global role administrators to and from a role. A role administrator can be a user or a role and does not require the MANAGE ROLES system privilege to administer a role.

You can appoint role administrators to a role during the creation process or after the role has been created and indicate whether they are also to be members of the role. If you do not specify any administrators, the global role administrator becomes the default administrator of the role.

If at least one role administrator is specified during role creation, global role administrators will be unable to manage the role because the SYS_MANAGE_ROLES_ROLE system privilege is not automatically granted to the role. For this reason, it is strongly recommended that you either do not define any role administrators when creating a role (add them after creation), or explicitly grant the SYS_MANAGE_ROLES_ROLE system privilege with administrative rights only along with any role administrators during the creation process.

If no role administrator is specified during the creation process, the global role administrator (SYS_MANAGE_ROLES_ROLE system privilege) is automatically granted to the role with administrative only rights.

If role administrators are later added to a role originally created with no role administrators specified, the global role administrator (SYS_MANAGE_ROLES_ROLE system privilege) may or may not be removed, depending on how the role administrators are added. If the GRANT statement is used, the SYS_MANAGE_ROLES_ROLE system privilege remains granted to the role. However, if the CREATE OR REPLACE statement is used, the SYS_MANAGE_ROLES_ROLE system privilege is removed if it is not explicitly included in the new list of role administrators.

**Note:** You will be unable to remove the SYS_MANAGE_ROLES_ROLE system privilege from a role if so doing would result in a failure to meet the minimum number of role administrators defined.

By default, the SYS_MANAGE_ROLES_ROLE system privilege is not granted to compatibility roles (SYS_AUTH_*_ROLE). Therefore, to allow global role administrators to manage a compatibility role, you must explicitly grant SYS_MANAGE_ROLES_ROLE with administrative rights only to the role.
**Adding a Role Administrator when Creating a Role**

Specify a role administrator when creating a new role.

**Prerequisites**
Requires the MANAGE ROLES system privilege.

**Task**
If at least one role administrator is specified during creation, global role administrators will be unable to manage the role unless explicitly specified.

For this reason, it is strongly recommended that you consider always adding the global role administrator to the list of role administrators during the creation process.

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

<table>
<thead>
<tr>
<th>Create Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative rights only; no role membership</td>
<td>CREATE ROLE <em>role_name</em> WITH ADMIN ONLY <em>admin_name</em> [,...]</td>
</tr>
<tr>
<td>Role and global role administrators granted</td>
<td>CREATE ROLE <em>role_name</em> WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, <em>admin_name</em> [,...]</td>
</tr>
<tr>
<td>Administrative rights along with role membership</td>
<td>CREATE ROLE <em>role_name</em> WITH ADMIN <em>admin_name</em> [,...]</td>
</tr>
</tbody>
</table>

*Since global role administrators cannot be granted membership in a role, you cannot include SYS_MANAGE_ROLES_ROLE in the administrators list when creating a role with role administrators granted membership in the role (WITH ADMIN option).*

**Example:**

Execute this statement to make Joe and Bob role administrators of the Sales role:

```
CREATE ROLE Sales WITH ADMIN Joe, Bob
```

Because it uses the WITH ADMIN clause, both Joe and Bob can both grant and revoke the role, as well as use the underlying system privileges of the role. If the WITH ADMIN ONLY clause were used, both Joe and Bob would be able to only grant and revoke the role.

Execute this statement to make Joe and Bob role administrators of the Sales role as well as allow global role administrators to manage the role:

```
CREATE ROLE Sales WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, Joe, Bob
```
Adding the Global Role Administrator when Creating a Role
Allow global role administrators to administer a new role.

Prerequisites
Requires the MANAGE ROLES system privilege.

Task
If at least one role administrator is specified during creation, global role administrators will be unable to manage the role unless explicitly specified.

For this reason, it is strongly recommended that you consider always adding the global role administrator to the list of role administrators during the creation process.

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

<table>
<thead>
<tr>
<th>Create Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global role administrator only; no role</td>
<td>CREATE ROLE role_name</td>
</tr>
<tr>
<td>administrators</td>
<td></td>
</tr>
<tr>
<td>Both role and global role administrators</td>
<td>CREATE ROLE role_name</td>
</tr>
<tr>
<td></td>
<td>WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, admin_name [,...]</td>
</tr>
</tbody>
</table>

*Global role administrator can only have administrative rights (WITH ADMIN ONLY) on a role. Therefore, when specifying both role and global role administrators when creating a new role, administrators can be granted with administrative rights only.

Example:
Execute this statement to create the Sales role and allow only global role administrators to manage it:

```
CREATE ROLE Sales
```

Execute this statement to make Joe and Bob role administrators of the Sales role, with administrative rights only, as well as allow global role administrators to manage the role:

```
CREATE ROLE Sales WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, Joe, Bob
```
**Adding Role Administrators to an Existing Role**
Add role administrators to an existing role. There is no maximum number of role administrators that can be granted to a single role.

**Prerequisites**
Requires administrative privilege over the role, or the MANAGE ROLES system privilege, if the role has a global role administrator.

**Task**
To add role administrators, execute one of these statements:

<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative privileges only</td>
<td>GRANT ROLE <em>role_name</em> TO <em>admin_name</em> [,...] WITH ADMIN ONLY OPTION</td>
</tr>
<tr>
<td>Administrative privileges and role membership</td>
<td>GRANT ROLE <em>role_name</em> TO <em>admin_name</em> [,...] WITH ADMIN OPTION</td>
</tr>
</tbody>
</table>

**Example:**
Execute this statement to make Mary and Bob role administrators of the Sales role.

```
GRANT ROLE Sales TO Mary, Bob WITH ADMIN ONLY OPTION
```

Each can administer the role, but not use its underlying system privileges because of the WITH ADMIN ONLY OPTION clause.

Execute this statement to make Sarah a role administrator of the Sales role with the ability to both administer the role and use its underlying system privileges because of the WITH ADMIN OPTION clause.

```
GRANT ROLE Sales TO Sarah WITH ADMIN OPTION
```

**See also**
- *GRANT ROLE Statement* on page 267

**Adding the Global Role Administrator to an Existing Role**
Add the global role administrator to an existing role.

**Prerequisites**
Requires administrative privilege over the role.
Task
The global role administrator can be granted to a role with administrative rights only (WITH
ADMIN ONLY option).
To reinstate the global role administrator on a role, execute:

```
GRANT ROLE role_name TO SYS_MANAGE_ROLES_ROLE
WITH ADMIN ONLY OPTION
```

See also
•  `GRANT ROLE Statement on page 267`

Making a User or Role a Global Role Administrator
Allow a user or role to act as a global role administrator.

Prerequisites
Requires the MANAGE ROLES system privilege granted with administrative rights.

Task
To become a global role administrator, you must be granted the MANAGE ROLES system
privilege. Administrative rights on the MANAGE ROLES system privilege are not required to
act as a global role administrator. If the MANAGE ROLES system privilege is granted to a
role, all members of the role inherit the system privilege, and thus the ability to act as a global
role administrator.
To grant the MANAGE ROLES system privilege execute this statement:

```
GRANT MANAGE ROLES TO grantee [, ...]
```

See also
•  `GRANT System Privilege Statement on page 274`

Replacing Existing Role Administrators on a Role
Replace current role administrators with new administrators.

Prerequisites
Requires administrative privilege over the role, or the MANAGE ROLES system privilege, if
the role has a global role administrator.

Task
Replacing role administrators can involve changing the users and roles who can act as
administrators and their level of administrative rights on the role. Depending on the extent of
the replacement, there are two approaches that can be taken. Each approach handles the
replacement task very differently, and have very different net effects on role and global
administrators. The first approach allows you to selectively replace the administrators of an
existing role. The second approach allows you to completely replace all existing role
administrators with new role administrators. It is important to note that replacement of administrators using the second approach includes the global role administrator.

The first approach is a two step process. It involves adding new role administrators and removing existing administrators from the role. Since the minimum number of administrators requirement must be met at all times through the process, it is recommended that you add before you remove. With this approach, if the role has a global role administrator, it is retained unless it is explicitly removed.

The second approach is a one step process, but has a much broader impact. It involves defining a new list of role administrators. All current role administrators are overwritten with new role administrators. If any current role administrators are to continue in this capacity, you must include them in the list of replacement role administrators. The list replaces all existing administrators, with the following behavior:

- All existing role administrators granted the WITH ADMIN OPTION not included on the new role administrators list become members of the role with no administrative rights on the role.
- All existing role administrators granted the WITH ADMIN ONLY OPTION not included on the new role administrators list are removed as members of the role.
- An existing role administrator included on the new role administrators list retains his or her original administrative rights if they are higher than the replacement rights. For example, if the new role administrators are granted WITH ADMIN ONLY rights, and UserA (an existing role administrator who was originally granted the role with WITH ADMIN rights) is included on the new list, UserA retains the higher WITH ADMIN rights.
- If the role has a global role administrator, it is removed from the role unless it is explicitly included on the new role administrators list.
- If the role has a global role administrator, and the new role administrators are granted WITH ADMIN rights, the global role administrator cannot be included in the list, since it cannot be granted WITH ADMIN rights. However, since it is not included on the list, it is removed from the role.

The replacement role command can be issued as long as the replacement administrative option is equal to or higher than the current level. To lower the administrative level, all role administrators must first be removed (revoked) from the role and then be regranted.

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 would violate the minimum requirement, the replacement fails.

To replace role administrators, execute one of these statements:
### Replacement Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace select role administrators with administrative only rights;</td>
<td>• GRANT ROLE role_name TO admin_name [,...] WITH ADMIN ONLY OPTION</td>
</tr>
<tr>
<td>no role membership</td>
<td>• REVOKE ADMIN OPTION FOR ROLE role_name FROM admin_name [,...]</td>
</tr>
<tr>
<td>Replace select role administrators with administrative and role</td>
<td>• GRANT ROLE role_name TO admin_name [,...] WITH ADMIN OPTION</td>
</tr>
<tr>
<td>membership</td>
<td>• REVOKE ADMIN OPTION FOR ROLE role_name FROM admin_name [,...]</td>
</tr>
<tr>
<td>Replace all role administrators with administrative rights only; no</td>
<td>CREATE OR REPLACE ROLE role_name WITH ADMIN ONLY admin_name [,...]</td>
</tr>
<tr>
<td>role membership. Remove the global role administrator, if exists.</td>
<td></td>
</tr>
<tr>
<td>Replace all role administrators with administrative rights and role</td>
<td>CREATE OR REPLACE ROLE role_name WITH ADMIN admin_name [,...]</td>
</tr>
<tr>
<td>membership. Remove the global role administrator, if exists.</td>
<td></td>
</tr>
<tr>
<td>Replace all role administrators with administrative rights only,</td>
<td>CREATE OR REPLACE ROLE role_name WITH ADMIN ONLY SYS_MANAGE_ROLES_ROLE, admin_name [,...]</td>
</tr>
<tr>
<td>including the global role administrator.*</td>
<td></td>
</tr>
<tr>
<td>Replace all role administrators with full administrative rights.</td>
<td>• CREATE OR REPLACE ROLE role_name WITH ADMIN admin_name [,...]</td>
</tr>
<tr>
<td>Restore the global role administrator to the role*</td>
<td>• GRANT ROLE role_name TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION</td>
</tr>
</tbody>
</table>

*SYS_MANAGE_ROLES_ROLE can only be granted to a role 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, a separate grant statement is required to grant SYS_MANAGE_ROLES_ROLE to the role using the WITH ADMIN ONLY option.

Examples:

Sales has Mary and Bob as role administrators with full administrative rights. Sales has a global role administrator.

Execute these statements to remove Bob as a role administrator and replace him with Sarah and Jeff, with the same administrative rights. Bob remains a member of Sales with no administrative rights.

```
GRANT ROLE sales TO Sarah, Jeff WITH ADMIN OPTION
REVOKE ADMIN OPTION FOR ROLE Sales FROM Bob
```

Execute these statements to replace the existing role administrators (Mary and Bob) with Sarah and Jeff, with full administrative rights. Since the global role administrator cannot be included on the list (cannot be granted with full administrative rights), it must be explicitly regranted to the role after replacement of the role administrators.

```
CREATE OR REPLACE ROLE Sales WITH ADMIN Sarah, Jeff
GRANT ROLE sales TO SYS_MANAGE_ROLES_ROLE WITH ADMIN ONLY OPTION
```

Execute these statements to replace the existing role administrators (Mary and Bob) with Bob and Sarah with administrative rights only. To preserve the global role administrator, it must be included on the list. Since Bob is to remain as a role administrator, and originally had higher administrative rights than the new role administrators, he retains the original higher administrative rights.

```
CREATE OR REPLACE ROLE Sales WITH ADMIN ONLY Bob, Sarah,
SYS_MANAGE_ROLES_ROLE
```

See also

- `GRANT ROLE Statement` on page 267
- `REVOKE ROLE Statement` on page 282
- `CREATE ROLE Statement` on page 254

Removing a Role Administrator from a Role

Remove a role administrator from a role.

Prerequisites

Requires administrative privilege over the role.

Task

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. You can remove role administrators only as long as the this minimum is still met after removal.
When removing a role administrator, if role administration was originally granted to the user using the WITH ADMIN OPTION clause, revoking role administration only removes their ability to manage the role (grant, revoke, drop), not the ability to use the underlying system privileges of the role (membership). However, if role administration was originally granted to the user using the WITH ADMIN ONLY OPTION clause, revoking role administration has the same effect as revoking the role entirely, as there was no membership associated with the role.

To remove a role administrator from a role, execute one of these statements:

<table>
<thead>
<tr>
<th>Removal Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove role administrator, but retain membership in the role.</td>
<td>REVOKE ADMIN OPTION FOR ROLE role_name FROM admin_name [,...]</td>
</tr>
<tr>
<td>Remove role administrator along with membership in the role.</td>
<td>REVOKE ROLE role_name FROM admin_name [,...]</td>
</tr>
</tbody>
</table>

**Example:**

This example assumes that both Mary and Sarah are currently role administrators of the Sales role. Mary has been granted both membership in the role and the ability to administer the role. Sarah, however, has only been granted the ability to administer the role, not membership. Due to the different administration levels granted, executing this statement to revoke administrative rights from the Sales role has a different impact on each administrator:

```
REVOKE ADMIN OPTION FOR ROLE Sales FROM Mary, Sarah
```

For Mary, this statement results in the loss of her ability to administer the Sales role, but retains her membership of the role. For Sarah, this statement revokes the Sales role completely from her.

**See also**
- *REVOKE ROLE Statement* on page 282

*Removing the Global Role Administrator from a Role*

Remove the global role administrator from a role.

**Prerequisites**

Requires administrative privilege over the role.

**Task**

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. You can remove the global role administrator from a role as long as this minimum is still met for the role.

To remove the global role administrator from a role, execute:

\begin{verbatim}
REVOKE ROLE role_name
FROM SYS_MANAGE_ROLES_ROLE
\end{verbatim}

See also

- \textit{REVOKE ROLE Statement} on page 282

\textbf{Minimum Number of Role Administrators}

The Minimum Number of Role Administrators (\texttt{MIN_ROLE_ADMINS}) option is a configurable value that ensures when dropping roles or users, you never create a scenario where there are no users and roles left with sufficient system privilege to manage the remaining users and roles.

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

- Creating roles
- Revoking roles
- Dropping users or roles
- Changing a user's password to null

\textbf{Note:} Users or roles without passwords cannot be administrators.

When you attempt to change this 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 would drop below the designated minimum value, the statement fails.

\textbf{Note:} Locked accounts are not considered when counting the number of administrators for a role.

\textbf{Example 1}

\begin{verbatim}
MIN_ROLE_ADMINS = 2
\end{verbatim}

Role1 has two administrators and Role2 has three administrators.

If you attempt to reduce the \texttt{min_role_admins} value to 1, the command succeeds because both roles still have the new designated minimum number of role administrators. However, if you attempt to increase the value to 3, the command fails because Role1 would no longer have sufficient administrators to meet the new minimum value.

\textbf{Example 2}

\begin{verbatim}
MIN_ROLE_ADMINS = 4
\end{verbatim}
Role1 has six administrators and Role2 has four administrators.

If you attempt to drop a user from Role1, the command succeeds because Role1 still has sufficient administrators to meet the minimum value. However, if you attempt to drop a user from Role2, the command fails because Role2 would no longer have sufficient administrators to meet the minimum value.

See also
- Automatic Unlocking of User Accounts on page 127
- MIN_ROLE_ADMINS Option on page 298

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

Prerequisites
Requires the SET ANY SECURITY OPTION system privilege to set this option.

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

This value applies to each role, not all roles in total. For example, if there are 20 roles and the minimum number of role administrators is set to two, each of the 20 roles must have a minimum of two role administrators defined, not two role administrators defined to administer the 20 roles in total.

To change the minimum number of role administrators, execute:

SET OPTION Public.min_role_admins = value

See also
- Automatic Unlocking of User Accounts on page 127
- MIN_ROLE_ADMINS Option on page 298

DBA User Unable to Administer a Role
Under certain circumstances, it is possible that the DBA user is unable to manage a role.

If the DBA user is unable to manage a role (grant, revoke, or drop a role), it is because:

- The global role administrator has been removed from the role; or
- The DBA user is not defined as a role administrator for the role.

To resolve the issue, grant the global role administrator to the role (recommended) or add the DBA user as a role administrator for the role.
Security Management

See also

- *GRANT ROLE Statement* on page 267
- *Adding Role Administrators to an Existing Role* on page 14
- *Adding the Global Role Administrator to an Existing Role* on page 14

System Roles

System roles are built-in roles that are automatically created in each newly created database.

System roles:

- Are automatically created in a new database.
- Cannot be dropped.
- Cannot have their default underlying system privileges modified or revoked.
- Additional roles and system privileges can be granted to and revoked from a system role.
- Cannot be granted with administrative rights (WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses).
- Have no password assigned, so it is not possible to connect to the database as a grantable system role.
- With the exception of the SYS, dbo, and rs_systabgroup role, do not own objects.

Granting dbo System Role

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

Prerequisites

Requires the MANAGE ROLES system privilege.

Task

The dbo system role is a member of the SYS system role. The PUBLIC system role is a member of the dbo system role. The dbo system role can be granted to other roles with no administrative rights only.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the dbo system role.

To grant the dbo system role, execute:

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

See also

- *GRANT ROLE Statement* on page 267
Granting diagnostics System Role
Members of the diagnostics role inherit SELECT, INSERT, UPDATE, DELETE, and ALTER privileges on diagnostic tables and views.

Prerequisites
Requires the MANAGE ROLES system privilege.

Task
The diagnostics system role is granted the MANAGE PROFILING system privilege with no administrative rights. By default, the diagnostics system role is granted to the SYS_AUTH_PROFILE_ROLE compatibility role with no administrative rights, and can be granted to other roles with no administrative rights only.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the diagnostics system role.

To grant the diagnostics system role, execute:
GRANT ROLE diagnostics TO grantee [, ...]

See also
• GRANT ROLE Statement on page 267

Granting PUBLIC System Role
The PUBLIC system role has SELECT permission on the system tables.

Prerequisites
Requires the MANAGE ROLES system privilege.

Task
The PUBLIC system role is a member of the SYS system role, and has read access for some of the system tables and views, so any user of the database can find out information about the database schema. If you want to restrict this access, you can 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 permissions specifically granted to that role.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the diagnostics system role.

To grant the PUBLIC system role, execute:
GRANT ROLE PUBLIC TO grantee [, ...]
See also
• *GRANT ROLE Statement* on page 267

**Granting rs_systabgroup System Role**
This role owns tables and system procedures required for Replication Server and grants users the underlying system privileges to perform replication server functionality.

**Prerequisites**
Requires the MANAGE ROLES system privilege.

**Task**
The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the rs_systabgroup system role.
To grant the rs_systabgroup system role, execute:

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

See also
• *GRANT ROLE Statement* on page 267

**Granting SYS System Role**
The SYS role owns the system tables and views for the database, which contain the full description of database schema, including all database objects and all user IDs.

**Prerequisites**
Requires the MANAGE ROLES system privilege.

**Task**
The SYS system role can be granted to other roles with no administrative rights only.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the SYS system role.
To grant the SYS system role, execute:

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

See also
• *GRANT ROLE Statement* on page 267
Granting SYS_REPLICATION_ADMIN_ROLE
This role is required for performing administration tasks related to replication such as granting replication roles, managing publications, subscriptions, synchronization users and profiles, managing message types, setting replication-related options, and so on.

Prerequisites
Requires the MANAGE ROLES system privilege.

Task
The SYS_REPLICATION_ADMIN_ROLE role is granted these system privileges with no administrative rights:

• CREATE ANY PROCEDURE
• CREATE ANY TABLE
• DROP ANY TABLE
• DROP ANY PROCEDURE
• MANAGE ANY OBJECT PRIVILEGE
• MANAGE ANY USER
• MANAGE ANY WEB SERVICE
• MANAGE REPLICATION
• MANAGE ROLES
• SERVER OPERATOR
• SELECT ANY TABLE
• SET ANY SYSTEM OPTION
• SET ANY PUBLIC OPTION
• SET ANY USER DEFINED OPTION

This default set of system privileges granted cannot be revoked from the role. However, unlike other system roles, additional system privileges and roles can be granted and revoked from this role.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the SYS_REPLICATION_ADMIN_ROLE system role.

To grant the SYS_REPLICATION_ADMIN_ROLE system role, execute:

GRANT ROLE SYS_REPLICATION_ADMIN_ROLE TO grantee [, ...]

See also
• GRANT ROLE Statement on page 267
Granting SYS_RUN_REPLICATION_ROLE
This role is required for performing replication tasks using dbremote and synchronization tasks using dbmlsync.

Prerequisites
MANAGE REPLICATION system privilege.

Task
The SYS_RUN_REPLICATION_ROLE system role is active only for users connecting through the dbremote or dbmlsync utilities.

The SYS_RUN_REPLICATION_ROLE system role is granted the SYS_AUTH_DBA_ROLE compatibility role with the WITH ADMIN OPTION clause. It is also granted these system privileges with the WITH NO ADMIN OPTION clause.

- SELECT ANY TABLE
- SET ANY USER DEFINED OPTION
- SET ANY SYSTEM OPTION
- BACKUP DATABASE
- MONITOR

By default, when granting SYS_RUN_REPLICATION_ROLE, the underlying system privileges were inherited by members of the receiving group. To prevent inheritance, the WITH NO SYSTEM PRIVILEGE INHERITANCE clause can be included for this system role only.

This default set of system privileges cannot be revoked from the system role. Additional system privileges and roles can be granted and revoked from this system role.

The minimum number of role administrators (MIN_ROLE_ADMINS) database option ensures that a designated number of users always exist in the database who can grant and revoke the MANAGE REPLICATION system privilege to other users.

The SYS_AUTH_DBA_ROLE compatibility role is granted by default to the SYS_RUN_REPLICATION_ROLE system role to address any possible requirements for additional system privileges to perform other replication related authorized tasks over and above the above-noted explicitly granted system privileges. It is recommended, however, that the SYS_AUTH_DBA_ROLE compatibility role be revoked from SYS_RUN_REPLICATION_ROLE system role and those specific additional system privileges or roles identified be explicitly granted to the SYS_RUN_REPLICATION_ROLE system role.

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the SYS_RUN_REPLICATION_ROLE system role.

To grant the SYS_RUN_REPLICATION_ROLE system role, execute one of these statements:
Inheritance Type | Statement
---|---
With inheritance | GRANT ROLE SYS_RUN_REPLICATION_ROLE TO *grantee* [, ...]
With no inheritance | GRANT ROLE SYS_RUN_REPLICATION_ROLE TO *grantee* [, ...]
| WITH NO SYSTEM PRIVILEGE INHERITANCE

See also
- *GRANT ROLE Statement* on page 267

**Granting 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 is the owner of all spatial objects.

**Prerequisites**
Requires the MANAGE ROLES system privilege.

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

The WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clauses are not valid when granting the SYS_SPATIAL_ADMIN_ROLE system role.

To grant the SYS_SPATIAL_ADMIN_ROLE system role, execute:

```
GRANT ROLE SYS_SPATIAL_ADMIN_ROLE TO *grantee* [, ...]
```

See also
- *GRANT ROLE Statement* on page 267

**Revoking a System Role**
Revoke a system role from a user or role.

**Prerequisites**
Requires administrative privilege over the system role being revoked.

**Task**
To revoke a system role, execute one of these statements:
## Administrative Option

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative rights</td>
<td><code>REVOKE ADMIN OPTION FOR ROLE role_name</code></td>
</tr>
<tr>
<td>to the role only</td>
<td><code>FROM grantee [...]</code></td>
</tr>
<tr>
<td>Role membership and all administrative</td>
<td><code>REVOKE ROLE role_name</code></td>
</tr>
<tr>
<td>rights to the role</td>
<td><code>FROM grantee [...]</code></td>
</tr>
</tbody>
</table>

### Examples:

This statement revokes SYS_AUTH_SA_ROLE entirely from Mary:

```
REVOKE ROLE SYS_AUTH_SA_ROLE FROM Mary
```

This statement revokes only administrative rights for SYS_AUTH_SSO_ROLE from Joe:

```
REVOKE ADMIN OPTION FOR ROLE SYS_AUTH_SSO_ROLE FROM Mary
```

### See also

- *REVOKE ROLE Statement* on page 282

## Compatibility Roles

Compatibility roles are like starter roles. They are also present for backward compatibility with pre-16.0 versions that support authority-based security.

You cannot modify the underlying system privileges of compatibility roles. However, you can migrate them to user-defined roles, and then modify the privileges. When you migrate a compatibility role, all grantees of the compatibility role are automatically granted the user-defined role.

For more information on compatibility roles, see *Upgrading to Role-Based Security* in the Migration document appropriate to your operating system.

## Granting SYS_AUTH_SA_ROLE

Allows users to perform authorized tasks pertaining to data and system administration responsibilities.

### Prerequisites

Administrative privilege over SYS_AUTH_SA_ROLE role.

### Task

You can grant this role with or without administrative rights. When granted with administrative rights, a user can manage (grant and revoke) the role, as well as use any of the underlying system privileges. When granted with administrative rights only, a user can
manage the role, but not use its underlying system privileges. Finally, when granted with no administrative rights, a user can only use its underlying system privileges.

To grant the SYS_AUTH_SANITIZE_ROLE role, execute one of these statements:

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>With full administrative rights</td>
<td><code>GRANT ROLE SYS_AUTH_SANITIZE_ROLE TO grantee [, ...] WITH ADMIN OPTION</code></td>
</tr>
<tr>
<td>With administrative rights only</td>
<td><code>GRANT ROLE SYS_AUTH_SANITIZE_ROLE TO grantee [, ...] WITH ADMIN ONLY OPTION</code></td>
</tr>
<tr>
<td>With no administrative rights</td>
<td><code>GRANT ROLE SYS_AUTH_SANITIZE_ROLE TO grantee [, ...] WITH NO ADMIN OPTION</code></td>
</tr>
</tbody>
</table>

See also

- `GRANT ROLE Statement` on page 267

**System Privileges Granted to SYS_AUTH_SANITIZE_ROLE**

System privileges granted to the SYS_AUTH_SANITIZE_ROLE role. Each system privilege is granted with the `WITH ADMIN OPTION` clause.

- ACCESS SERVER LS system privilege
- ALTER ANY INDEX system privilege
- ALTER ANY MATERIALIZED VIEW system privilege
- ALTER ANY OBJECT system privilege
- ALTER ANY PROCEDURE system privilege
- ALTER ANY SEQUENCE system privilege
- ALTER ANY TEXT CONFIGURATION system privilege
- ALTER ANY TABLE system privilege
- ALTER ANY TRIGGER system privilege
- ALTER ANY VIEW system privilege
- ALTER DATABASE system privilege
- ALTER DATATYPE system privilege
- BACKUP DATABASE system privilege
- CHECKPOINT system privilege
- COMMENT ANY OBJECT system privilege
- CREATE ANY INDEX system privilege
- CREATE ANY MATERIALIZED VIEW system privilege
- CREATE ANY OBJECT system privilege
- CREATE ANY PROCEDURE system privilege
• CREATE ANY SEQUENCE system privilege
• CREATE ANY TABLE system privilege
• CREATE ANY TEXT CONFIGURATION system privilege
• CREATE ANY TRIGGER system privilege
• CREATE ANY VIEW system privilege
• CREATE DATATYPE system privilege
• CREATE EXTERNAL REFERENCE system privilege
• CREATE MATERIALIZED VIEW system privilege
• CREATE MESSAGE system privilege
• CREATE PROCEDURE system privilege
• CREATE PROXY TABLE system privilege
• CREATE TABLE system privilege
• CREATE TEXT CONFIGURATION system privilege
• CREATE VIEW system privilege
• DEBUG ANY PROCEDURE system privilege
• DELETE ANY TABLE system privilege
• DROP ANY INDEX system privilege
• DROP ANY MATERIALIZED VIEW system privilege
• DROP ANY OBJECT system privilege
• DROP ANY PROCEDURE system privilege
• DROP ANY SEQUENCE system privilege
• DROP ANY TABLE system privilege
• DROP ANY TEXT CONFIGURATION system privilege
• DROP ANY VIEW system privilege
• DROP DATATYPE system privilege
• DROP MESSAGE system privilege
• EXECUTE ANY PROCEDURE system privilege
• INSERT ANY TABLE system privilege
• LOAD ANY TABLE system privilege
• MANAGE ANY DBSPACE system privilege
• MANAGE ANY EVENT system privilege
• MANAGE ANY EXTERNAL ENVIRONMENT system privilege
• MANAGE ANY EXTERNAL OBJECT system privilege
• MANAGE ANY MIRROR SERVER system privilege
• MANAGE ANY SPATIAL OBJECT system privilege
• MANAGE ANY STATISTICS system privilege
• MANAGE ANY WEB SERVICE system privilege
• MANAGE MULTIPLEX system privilege
• MANAGE PROFILING system privilege
• MANAGE REPLICATION system privilege
• MONITOR system privilege
• READ CLIENT FILE system privilege
• READ FILE system privilege
• REORGANIZE ANY OBJECT system privilege
• SELECT ANY TABLE system privilege
• SERVER OPERATOR system privilege
• SET ANY PUBLIC OPTION system privilege
• SET ANY SYSTEM OPTION system privilege
• SET ANY USER DEFINED OPTION system privilege
• TRUNCATE ANY TABLE system privilege
• UPDATE ANY TABLE system privilege
• UPGRADE ROLE system privilege
• USE ANY SEQUENCE system privilege
• VALIDATE ANY OBJECT system privilege
• WRITE CLIENT FILE system privilege
• WRITE FILE system privilege

**Granting SYS_AUTH_SSO_ROLE**

Grant to allow users to perform authorized tasks pertaining to security and access control responsibilities.

**Prerequisites**

Administrative privilege over SYS_AUTH_SSO_ROLE role.

**Task**

You can grant this role with or without administrative rights. When granted with administrative rights, a user can manage (grant and revoke) the role, as well as use any of the underlying system privileges. When granted with administrative rights only, a user can manage the role, but not use its underlying system privileges. Finally, when granted with no administrative rights, a user can only use its underlying system privileges.

To grant the role, execute one of these statements:

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>With full administrative rights</td>
<td>GRANT ROLE SYS_AUTH_SSO_ROLE TO grantees [....] WITH ADMIN OPTION</td>
</tr>
<tr>
<td>With administrative rights only</td>
<td>GRANT ROLE SYS_AUTH_SSO_ROLE TO grantees [....] WITH ADMIN ONLY OPTION</td>
</tr>
</tbody>
</table>
Administrative Option | Statement
--- | ---
With no administrative rights | GRANT ROLE SYS_AUTH_SSO_ROLE TO grantee [,...]
 | WITH NO ADMIN OPTION

See also

- *GRANT ROLE Statement* on page 267

**System Privileges Granted to SYS_AUTH_SSO_ROLE**

System privileges granted to the SYS_AUTH_SSO_ROLE role. Each system privilege is granted with the *WITH ADMIN OPTION* clause.

- ALTER ANY OBJECT OWNER system privilege
- ANY USER system privilege
- CHANGE PASSWORD system privilege
- DROP CONNECTION system privilege
- MANAGE ANY OBJECT PRIVILEGES system privilege
- MANAGE ANY LDAP SERVER system privilege
- MANAGE ANY LOGIN POLICY system privilege
- MANAGE ANY USER system privilege
- MANAGE AUDITING system privilege
- MANAGE ROLES system privilege
- SET ANY SECURITY OPTION system privilege
- SET USER system privilege (granted with the WITH ADMIN ONLY OPTION clause)

**Granting SYS_AUTH_DBA_ROLE**

Grant to allow users to perform all authorized tasks.

**Prerequisites**

Administrative privilege over SYS_AUTH_DBA_ROLE role.

**Task**

This role indirectly grants all compatibility roles, as well as some system roles to a user. It is the union of the underlying system privileges of each of these roles that makes the SYS_AUTH_DBA_ROLE role the "super" role.

You can grant this role with or without administrative rights. When granted with administrative rights, a user can manage (grant and revoke) the role, as well as use any of the underlying system privileges. When granted with administrative rights only, a user can manage the role, but not use its underlying system privileges. Finally, when granted with no administrative rights, a user can only use its underlying system privileges.

**Note:** If you are migrating from SAP Sybase IQ 15.4 or earlier, the concept of inheritance of the underlying system privileges of this system role represents a change in behavior with SAP.
Sybase IQ 16.0 or later. For SAP Sybase IQ 15.4 and earlier behavior, use the WITH NO SYSTEM PRIVILEGE INHERITANCE clause.

The WITH ADMIN ONLY OPTION clauses is invalid when using the WITH NO SYSTEM PRIVILEGE INHERITANCE clause. The WITH NO ADMIN OPTION clause is valid, but not required, as it is semantically equivalent to the WITH NO SYSTEM PRIVILEGE INHERITANCE clause.

To grant the SYS_AUTH_DBA_ROLE role, execute one of these statements:

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>With full administrative rights</td>
<td>GRANT ROLE SYS_AUTH_DBA_ROLE TO grantee [,....] WITH ADMIN OPTION</td>
</tr>
<tr>
<td>With administrative rights only</td>
<td>GRANT ROLE SYS_AUTH_DBA_ROLE TO grantee [,....] WITH ADMIN ONLY OPTION</td>
</tr>
<tr>
<td>With no administrative rights</td>
<td>GRANT ROLE SYS_AUTH_DBA_ROLE TO grantee [,....] WITH NO ADMIN OPTION</td>
</tr>
<tr>
<td>With full administrative rights,</td>
<td>GRANT ROLE SYS_AUTH_REMOTE_DBA_ROLE TO user_ID \</td>
</tr>
<tr>
<td>but no system privilege inheritance</td>
<td>WITH ADMIN OPTION \</td>
</tr>
<tr>
<td></td>
<td>WITH NO SYSTEM PRIVILEGE INHERITANCE</td>
</tr>
</tbody>
</table>

See also
- *GRANT ROLE Statement* on page 267

Roles Granted to SYS_AUTH_DBA_ROLE

Roles granted to the SYS_AUTH_DBA_ROLE role.

These compatibility roles are granted with the WITH ADMIN OPTION clause:

- SYS_AUTH_SA_ROLE
- SYS_AUTH_SSO_ROLE

These compatibility roles are granted with the WITH ADMIN ONLY OPTION clause:

- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE
- SYS_AUTH_READFILE_ROLE
- SYS_AUTH_PROFILE_ROLE
• SYS_AUTH_READCLIENTFILE_ROLE
• SYS_AUTH_WRITECLIENTFILE_ROLE
• SYS_AUTH_WRITEFILE_ROLE
• SYS_AUTH_USER_ADMIN_ROLE
• SYS_AUTH_SPACE_ADMIN_ROLE
• SYS_AUTHMULTIPLEX_ADMIN_ROLE
• SYS_AUTH_OPERATOR_ROLE
• SYS_AUTH_PERMS_ADMIN_ROLE

These system roles are granted with the WITH ADMIN ONLY OPTION clause:

• SYS_SPATIAL_ADMIN_ROLE
• diagnostics
• rs_systabgroup
• SYS
• DBO
• PUBLIC

System Privileges Granted to SYS_AUTH_DBA_ROLE
System privileges granted to the SYS_AUTH_DBA_ROLE role.

Through the granting of all compatibility roles and select system roles, these system privileges are indirectly granted to the SYS_AUTH_DBA_ROLE role. The underlying system privileges of the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE roles are indirectly granted with the WITH ADMIN OPTION clause, which grants full administrative rights. All other compatibility roles and system roles are indirectly granted with the WITH ADMIN ONLY OPTION clause.

• ACCESS SERVER LS system privilege
• ALTER ANY INDEX system privilege
• ALTER ANY MATERIALIZED VIEW system privilege
• ALTER ANY OBJECT system privilege
• ALTER ANY OBJECT OWNER system privilege
• ALTER ANY PROCEDURE system privilege
• ALTER ANY SEQUENCE system privilege
• ALTER ANY TABLE system privilege
• ALTER ANY TEXT CONFIGURATION system privilege
• ALTER ANY TRIGGER system privilege
• ALTER ANY VIEW system privilege
• ALTER DATABASE system privilege
• ALTER DATATYPE system privilege
• BACKUP DATABASE system privilege
• CHANGE PASSWORD system privilege
• CHECKPOINT system privilege
• COMMENT ANY OBJECT system privilege
• CREATE ANY INDEX system privilege
• CREATE ANY MATERIALIZED VIEW system privilege
• CREATE ANY OBJECT system privilege
• CREATE ANY PROCEDURE system privilege
• CREATE ANY SEQUENCE system privilege
• CREATE ANY TABLE system privilege
• CREATE ANY TEXT CONFIGURATION system privilege
• CREATE ANY TRIGGER system privilege
• CREATE ANY VIEW system privilege
• CREATE DATATYPE system privilege
• CREATE EXTERNAL REFERENCE system privilege
• CREATE MATERIALIZED VIEW system privilege
• CREATE MESSAGE system privilege
• CREATE PROCEDURE system privilege
• CREATE PROXY TABLE system privilege
• CREATE TABLE system privilege
• CREATE TEXT CONFIGURATION system privilege
• CREATE VIEW system privilege
• DEBUG ANY PROCEDURE system privilege
• DELETE ANY TABLE system privilege
• DROP ANY INDEX system privilege
• DROP ANY MATERIALIZED VIEW system privilege
• DROP ANY OBJECT system privilege
• DROP ANY PROCEDURE system privilege
• DROP ANY SEQUENCE system privilege
• DROP ANY TABLE system privilege
• DROP ANY TEXT CONFIGURATION system privilege
• DROP ANY VIEW system privilege
• DROP CONNECTION system privilege
• DROP DATATYPE system privilege
• DROP MESSAGE system privilege
• EXECUTE ANY PROCEDURE system privilege
• LOAD ANY TABLE system privilege
• INSERT ANY TABLE system privilege
• MANAGE ANY DBSPACE system privilege
• MANAGE ANY EVENT system privilege
• MANAGE ANY EXTERNAL ENVIRONMENT system privilege
Revoking a Compatibility Role
Revoke a compatibility role from a user or role.

Prerequisites
Requires administrative privilege over the compatibility role being revoked.

Task
To revoke a compatibility role, execute one of these statements:
### Administrative Option

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative rights only</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>REVOKE ADMIN OPTION FOR ROLE</code></td>
</tr>
<tr>
<td></td>
<td><code>compatibility_role</code></td>
</tr>
<tr>
<td></td>
<td><code>FROM grantee [,...]</code></td>
</tr>
<tr>
<td>Membership in the role and</td>
<td><code>REVOKE ROLE</code></td>
</tr>
<tr>
<td>any administrative rights</td>
<td><code>compatibility_role</code></td>
</tr>
<tr>
<td></td>
<td><code>FROM grantee [,...]</code></td>
</tr>
</tbody>
</table>

### See also
- *REVOKE ROLE Statement* on page 282

### Views, Procedures and Tables That Are Owned by Roles

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

Make users who need access to a table, view, or stored procedure members of the role that owns the object. This eliminates the need to qualify the object name when accessing.

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.

System privileges should not be granted 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.

Display Roles Granted

The sp_displayroles stored procedure which returns all roles granted to the specified system privilege, system role, user-defined role, or user name, or displays the entire hierarchy tree of roles.

The report includes role name, parent role name, type of grant (with or without administrative privilege) and the level of the role hierarchy.

No system privileges are required to execute the procedure on your own user ID. To execute the procedure on other users requires the MANAGE ROLES system privilege. To execute the procedure for a role or system privilege requires administrative privilege over the role or system privilege specified.

Example

The following statement returns all roles granted to the user issuing the command. In this example, the user logged has been granted the SYS_AUTH_DBA_ROLE compatibility role with administrative rights (for example, GRANT ROLE SYS_AUTH_DBA_ROLE TO User1 WITH ADMIN OPTION;).

CALL sp_displayroles();

This examples returns the list of system privileges granted to the SYS_SPATIAL_ADMIN_ROLE system role:

CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE' );

<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANAGE ANY SPATIAL OBJECT</td>
<td>(NULL)</td>
<td>NO ADMIN</td>
<td>1</td>
</tr>
</tbody>
</table>

This examples returns the list of system privileges granted to the SYS_SPATIAL_ADMIN_ROLE, including all roles above it in the hierarchy of roles:

CALL sp_displayroles( 'SYS_SPATIAL_ADMIN_ROLE', 'expand_up');
<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_AUTH_DBA_ROLE</td>
<td>dbo</td>
<td>ADMIN</td>
<td>-3</td>
</tr>
<tr>
<td>SYS_AUTH_SSO_ROLE</td>
<td>SYS_AUTH_DBA_ROLE</td>
<td>ADMIN</td>
<td>-3</td>
</tr>
<tr>
<td>MANAGE.Roles</td>
<td>SYS_AUTH_REMOTEDBA_ROLE</td>
<td>ADMIN</td>
<td>-2</td>
</tr>
<tr>
<td>MANAGE.Roles</td>
<td>SYS_AUTH_SSO_ROLE</td>
<td>ADMIN</td>
<td>-1</td>
</tr>
<tr>
<td>MANAGE.Roles</td>
<td>SYS_REPLICATIONADMIN_ROLE</td>
<td>NO ADMIN</td>
<td>-1</td>
</tr>
<tr>
<td>SYS_SPATIALADMIN_ROLE</td>
<td>MANAGE.ROLES</td>
<td>ADMIN</td>
<td>0</td>
</tr>
</tbody>
</table>

See also

• *sp_displayroles System Procedure* on page 317

**Determine Roles and Privileges Granted to a User**

The *sp_has_role* stored function returns an integer value which 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.

See also

• *SP_HAS_ROLE Function [System]* on page 320
Privileges

A privilege is a right to perform a privileged 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-level privileges.

System privileges give you the general right to perform a privileged operation, while object-level 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 system. If you have the ALTER TABLE system privilege, you can only alter tables you own, or tables on which you have been granted the ALTER object-level privilege. They can be granted or revoked, but not created or dropped.

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.

You grant and revoke system and object-level privileges by using the GRANT and REVOKE statements.

Privileges Versus Permissions

Permission and privilege do not mean the same thing in role-based security. A user may have been granted the privilege required to perform an authorized task, but not have the necessary permission to perform the authorized task on the required object.

A privilege grants a user or role the ability to perform a specific authorized task. Permission, however, refers to the context in which the task is being performed.

When performing an authorized task, if a failure occurs, the error message that appears often indicates that the user does not have permission to perform the task, not that the user does not have the privilege to perform the task. Before executing a privileged task or operation, the system verifies that the user has the correct privilege to perform the:

- privileged operation.
- privileged operation on the acted-on object
- privilege operation in the context they are trying to do it in.

If the user does not have the correct privilege at any level, he or she is said to not have permission to perform the task. The operation fails and an error message appears.
Example

A user has been granted the ALTER privilege only on a text configuration object called Myconfig.

Object privilege: The user attempts to alter a text configuration object other than Myconfig. The task fails because the ALTER privilege granted to the user is specific to the Myconfig text object, not any text object.

Context privilege: The user attempts to drop a prefilter on Myconfig. Though the user has been granted the ALTER privilege on Myconfig, to drop a prefilter on a text configuration object requires the ALTER ANY TEXT CONFIGURATION or ALTER ANY OBJECT system privilege, which has not been granted to the user.

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, is granted by default to either the SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE role, but not both. The exception, SET USER system privilege, is granted in both roles. Some select system privileges are also vested in other predefined system roles.

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

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

System Privileges Listed by Functional Area

A list of system privileges organized by functional area.

Database System Privileges

System privileges pertaining to performing authorized tasks on databases.

See also

- List All System Privileges on page 78
**ALTER DATABASE System Privilege**
Required to alter a database.

The ALTER DATABASE system privilege allows a user to:

- Perform a database upgrade
- Perform cost model calibration
- Load statistics
- Change transaction logs (also requires the SERVER OPERATOR system privilege)
- Change ownership of the database (also requires the MANAGE ANY MIRROR SERVER system privilege)

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**BACKUP DATABASE System Privilege**
Allows a user to back up a database on one or more archive devices.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**CHECKPOINT System Privilege**
Required to force the database server to execute a checkpoint.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78
**DROP CONNECTION System Privilege**
Required to drop any user connections to the database.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MANAGE PROFILING System Privilege**
Required to enable or disable server tracing for application profiling. The DIAGNOSTICS system role is also required to fully utilize diagnostics functionality for user information.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MONITOR System Privilege**
Required to allow a user to perform monitoring related tasks such as access privileged statistics, run server monitor related procedures, and so on.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Database Options System Privileges**
System privileges pertaining to performing authorized tasks to set database options.

**See also**
- *List All System Privileges* on page 78
**SET ANY PUBLIC OPTION System Privilege**

Required to set any PUBLIC system database option that does not require the SET ANY SECURITY OPTION or SET ANY SYSTEM OPTION system privileges.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**SET ANY SECURITY OPTION System Privilege**

Required to set any PUBLIC security database options.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**SET ANY SYSTEM OPTION System Privilege**

Required to set any PUBLIC system database options.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**SET ANY USER DEFINED OPTION System Privilege**

Required to set any user-defined options.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

Data Type System Privileges
System privileges pertaining to performing authorized tasks on data types.

See also
• List All System Privileges on page 78

ALTER DATATYPE System Privilege
Required to alter data types.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

CREATE DATATYPE System Privilege
Required to create data types.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

DROP DATATYPE System Privilege
Required to drop data types.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
Dbspaces System Privileges
System privileges pertaining to performing authorized tasks on dbspaces.

See also
- List All System Privileges on page 78

MANAGE ANY DBSPACE System Privilege
Required to perform management-related tasks on dbspaces.

The MANAGE ANY DBSPACE system privilege allows a user to:

- Issue CREATE, ALTER, DROP, or COMMENT statements on any dbspace
- GRANT or REVOKE the CREATE object-level privilege on any dbspace
- Move data to any dbspace
- Issue a read-only selective restore statement on any dbspace
- Run the database delete file function

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

Debugging System Privileges
System privileges pertaining to performing authorized tasks related to debugging.

See also
- List All System Privileges on page 78

DEBUG ANY PROCEDURE System Privilege
Required to debug all code in any database object.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
• \textit{REVOKE System Privilege Statement} on page 286
• \textit{List All System Privileges} on page 78

\textbf{Events System Privileges}
System privileges pertaining to authorized tasks on events.

\textbf{See also}
• \textit{List All System Privileges} on page 78

\textit{MANAGE ANY EVENT System Privilege}
Required to create, alter, drop or trigger events.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

\textbf{See also}
• \textit{GRANT System Privilege Statement} on page 274
• \textit{REVOKE System Privilege Statement} on page 286
• \textit{List All System Privileges} on page 78

\textbf{External Environment System Privileges}
System privileges pertaining to performing authorized tasks on external environments.

\textbf{See also}
• \textit{List All System Privileges} on page 78

\textit{CREATE EXTERNAL REFERENCE System Privilege}
Required to create external references in the database.

This system privilege is required in addition to any other system privileges required for creating a database object that references an external object.

For example:
• To create an external term breaker or a self-owned text configuration that uses an external term breaker requires the system privilege \text{CREATE TEXT CONFIGURATION} in addition to the \text{CREATE EXTERNAL REFERENCE} system privilege.
• To create an external procedure or function requires the \text{CREATE PROCEDURE} system privilege in addition to the \text{CREATE EXTERNAL REFERENCE} system privilege.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MANAGE ANY EXTERNAL ENVIRONMENT System Privilege**

Required to manage external environments.

The MANAGE ANY EXTERNAL ENVIRONMENT system privilege allows a user to:

- Issue ALTER or COMMENT statements on an external environment
- Issue START or STOP statements on an external environment

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MANAGE ANY EXTERNAL OBJECT System Privilege**

Required to issue install, comment on, or remove external objects.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Files System Privileges**

System privileges pertaining to authorized tasks for files.

See also

- *List All System Privileges* on page 78
**READ CLIENT FILE System Privilege**
Required to read a file resident on the client machine.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**READ FILE System Privilege**
Required to read a file resident on the server machine.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**WRITE CLIENT FILE System Privilege**
Required to write a file resident on the client machine.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**WRITE FILE System Privilege**
Required to write a file resident on the server machine.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
Indexes System Privileges
System privileges pertaining to authorized tasks for indexes.

See also
• List All System Privileges on page 78

ALTER ANY INDEX System Privilege
Required to alter an existing index.

The ALTER ANY INDEX system privilege allows a user to:
• Alter indexes on any table owned by any user
• Issue COMMENT statement on any index owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

CREATE ANY INDEX System Privilege
Required to create a new index.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

The CREATE ANY INDEX system privilege allows a user to:
• Create indexes on any table owned by any user
• Issue COMMENT statement on any index owned by any user

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78
**DROP ANY INDEX System Privilege**
Required to drop indexes on any table owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**LDAP System Privileges**
System privileges pertaining to performing authorized tasks on an LDAP server configuration object.

**See also**
- *List All System Privileges* on page 78

**MANAGE ANY LDAP SERVER System Privilege**
Required to issue CREATE, ALTER, or DROP statements on an LDAP server configuration object.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Materialized Views System Privileges**
System privileges pertaining to performing authorized tasks on materialized views.

**See also**
- *List All System Privileges* on page 78
CREATE ANY MATERIALIZED VIEW System Privilege
Required to create materialized views that are owned by any user. It also allows users to issue the COMMENT statement on materialized views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

CREATE MATERIALIZED VIEW System Privilege
Required to create self-owned materialized views. It also allows users to issue the COMMENT statement on self-owned materialized views.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

ALTER ANY MATERIALIZED VIEW System Privilege
Required to alter materialized views owned by any user. It also allows users to issue the COMMENT statement on materialized views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78
**DROP ANY MATERIALIZED VIEW System Privilege**
Required to drop materialized views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Messages System Privileges**
System privileges pertaining to performing authorized tasks for messages.

**See also**
- *List All System Privileges* on page 78

**CREATE MESSAGE System Privilege**
Required to create messages.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**DROP MESSAGE System Privilege**
Required to drop messages.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78
**Miscellaneous System Privileges**

System privileges pertaining to performing miscellaneous authorized tasks.

See also

- *List All System Privileges* on page 78

**ALTER ANY OBJECT System Privilege**

Required to alter an object owned by anyone.

The ALTER ANY OBJECT system privilege allows a user to issue these statements:

- ALTER TABLE
- ALTER INDEX
- ALTER JOIN INDEX
- ALTER VIEW
- ALTER MATERIALIZED VIEW
- ALTER PROCEDURE
- ALTER EVENT
- ALTER SEQUENCE
- ALTER FUNCTION
- ALTER DATATYPE
- ALTER MESSAGE
- ALTER TEXT CONFIGURATION
- ALTER TRIGGER
- ALTER STATISTICS
- COMMENT on different objects
- ALTER SPATIAL REFERENCE SYSTEM
- ALTER SPATIAL UNIT OF MEASURE

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78
**ALTER ANY OBJECT OWNER System Privilege**

Required to change the owner of a user table owned by anyone.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**Note:** This system privilege applies to table objects only. Owners of other objects, such as procedures, materialized views, etc., cannot be changed.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**COMMENT ANY OBJECT System Privilege**

Required to comment on any object owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**CREATE ANY OBJECT System Privilege**

Required to create an object owned by anyone.

The CREATE ANY OBJECT system privilege allows a user to issue these statements:

- COMMENT on different objects
- CREATE DATATYPE
- CREATE EVENT
- CREATE FUNCTION
- CREATE INDEX
- CREATE JOIN INDEX
- CREATE MATERIALIZED VIEW
- CREATE MESSAGE
- CREATE PROCEDURE
- CREATE SCHEMA
- CREATE SEQUENCE
• CREATE SPATIAL REFERENCE SYSTEM
• CREATE SPATIAL UNIT OF MEASURE
• CREATE STATISTICS
• CREATE TABLE
• CREATE TEXT CONFIGURATION
• CREATE VIEW

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

DROP ANY OBJECT System Privilege
Required to drop an object owned by anyone.

The DROP ANY OBJECT system privilege allows a user to issue these statements:

• DROP DATATYPE
• DROP EVENT
• DROP FUNCTION
• DROP INDEX
• DROP JOIN INDEX
• DROP MATERIALIZED VIEW
• DROP MESSAGE
• DROP PROCEDURE
• DROP SEQUENCE
• DROP SPATIAL REFERENCE SYSTEM
• DROP SPATIAL UNIT OF MEASURE
• DROP STATISTICS
• DROP TABLE
• DROP TEXT CONFIGURATION
• DROP TRIGGER
• DROP VIEW

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78

**MANAGE ANY OBJECT PRIVILEGES System Privilege**
Required to manage objects.

The MANAGE ANY OBJECT PRIVILEGES system privilege allows a user to perform management-related tasks such as:

- Grant any object-level privilege (INSERT, UPDATE, DELETE, SELECT, ALTER, REFERENCES or EXECUTE) on objects owned by any user
- Revoke any object-level privilege granted by the object owner or another user with MANAGE ANY OBJECT PRIVILEGES system privilege

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78

**REORGANIZE ANY OBJECT System Privilege**
Required to issue the REORGANIZE statement on applicable objects owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78

**VALIDATE ANY OBJECT System Privilege**
Required to validate or check tables, materialized views, indexes or databases in the system store owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Mirror Server System Privileges**
System privileges pertaining to authorized tasks for mirrored servers.

See also

- *List All System Privileges* on page 78

**MANAGE ANY MIRROR SERVER System Privilege**
Required to perform high availability server administrative tasks.

The MANAGE ANY MIRROR SERVER system privilege allows a user to:

- Issue CREATE, ALTER or DROP statement on mirrored servers
- Change mirror server parameters
- Set options on mirror servers
- Execute the ALTER statement to change ownership of a database

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Multiplex System Privileges**
Two specific system privileges are required to perform authorized tasks in a multiplex environment.

See also

- *List All System Privileges* on page 78

**ACCESS SERVER LS System Privilege**
Allows logical server connection using the SERVER logical server context.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MANAGE MULTIPLEX System Privilege**
Allows administrative tasks related to multiplex server management.

The MANAGE MULTIPLEX system privilege allows a user to:

- Issue multiplex-related CREATE, ALTER, DROP, or COMMENT statements on logical server policies
- Issue multiplex-related CREATE, ALTER, DROP, or COMMENT statements on logical servers
- Perform exclusive assignment of a dbspace to logical servers
- Release a populated dbspace from the exclusive use of a logical server

**Note:** The MANAGE MULTIPLEX system privilege also manages failover configurations, and is required for a manual failover.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Procedures System Privileges**
System privileges pertaining to performing authorized tasks for procedures.

See also

- *List All System Privileges* on page 78

**ALTER ANY PROCEDURE System Privilege**
Required to alter any stored procedure or function owned by any user.

The ALTER ANY PROCEDURE system privilege allows a user to:

- Alter stored procedures and functions owned by any user
- Issue COMMENT statement on procedures owned by any user
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

CREATE ANY PROCEDURE System Privilege

Required to create any stored procedure or function owned by any user.

The CREATE ANY PROCEDURE system privilege allows a user to:
- Create stored procedures and functions owned by any user
- Issue COMMENT statement on procedures owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

CREATE PROCEDURE System Privilege

Required to create a self-owned stored procedure or function.

The CREATE PROCEDURE system privilege allows a user to:
- Create self-owned stored procedures and functions
- Issue COMMENT statement on self-owned procedures

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78
DROP ANY PROCEDURE System Privilege
Required to drop any stored procedure or function owned by any user.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

EXECUTE ANY PROCEDURE System Privilege
System privilege required to execute any stored procedure or function owned by any user.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

MANAGE AUDITING System Privilege
Required to run the sa_audit_string stored procedure.
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

Replication System Privileges
System privileges pertaining to performing authorized replication tasks.

See also
- List All System Privileges on page 78
MANAGE REPLICA TION System Privilege
System privilege required to perform replication-related tasks.

The MANAGE REPLICA TION system privilege allows a user to:

- Issue CREATE, ALTER, DROP, or COMMENT PUBLICATION statement
- Issue CREATE, ALTER, DROP, or SYNCHRONIZATION SUBSCRIPTION statement
- Issue CREATE, ALTER, DROP, or SYNCHRONIZATION USER statement
- Issue CREATE, ALTER, DROP, or COMMENT SYNCHRONIZATION PROFILE statement
- Issue CREATE or DROP SUBSCRIPTION statement
- Issue CREATE REMOTE MESSAGE TYPE statement
- Issue DROP REMOTE MESSAGE TYPE statement
- Issue GRANT or REVOKE CONSOLIDATE statement
- Issue GRANT or REVOKE REMOTE statement
- Issue GRANT or REVOKE PUBLISH statement
- Issue LOCK FEATURE statement
- Issue START, STOP, or SYNCHRONIZE SUBSCRIPTION statement
- Issue PASSSTHROUGH statement
- Issue REMOTE RESET statement
- Issue SET REMOTE OPTION statement
- Issue START or STOP SYNCHRONIZATION SCHEMA CHANGE statement
- Issue SYNCHRONIZE PROFILE statement
- Execute SA_SETREMOTEUSER
- Execute SA_SETSUBSCRIPTION

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

Roles System Privileges
System privileges pertaining to performing authorized tasks for roles.

See also
- List All System Privileges on page 78
MANAGE ROLES System Privilege

Required to create new roles and act as the default administrator of roles.

While the MANAGE ROLES system privilege allows a user to create a new user-defined role, it does not allow them to delete the role. For this, a user requires administrative rights on the role.

Users granted the MANAGE ROLES system privilege serve as default global role administrators on a user-defined role.

If no role administrator is specified during the role creation process, the MANAGE ROLES system privilege (SYS_MANAGE_ROLES_ROLE) is automatically granted to the role with the ADMIN ONLY OPTION clause, which allows the global role administrator to administer the role. If at least one role administrator is specified during the creation process, the MANAGE ROLES system privilege is not granted to the role, and global role administrators will be unable to manage the role.

MANAGE ROLES is the only system privilege with the ability to be granted the ability to administer user-defined roles.

Note: Administration of a role can also be granted directly to users either during the creation of the role or after the fact. When granted directly to a user, the user does not require the MANAGE ROLES system privilege to administer the role.

See also

- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

UPGRADE ROLE System Privilege

Required to administrate new system privileges introduced when upgrading from a database earlier than 16.0.

By default, the UPGRADE ROLE system privilege is granted to the SYS_AUTH_SA_ROLE role (if it still exists).

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78
**Sequences System Privileges**
System privileges pertaining to performing authorized tasks for sequencing.

**See also**
- *List All System Privileges* on page 78

**ALTER ANY SEQUENCE System Privilege**
Required to alter any sequence.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**CREATE ANY SEQUENCE System Privilege**
Required to create any sequence.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**DROP ANY SEQUENCE System Privilege**
Required to drop any sequence.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78
USE ANY SEQUENCE System Privilege
Required to use any sequence.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

Server Operator System Privileges
System privileges pertaining to performing authorized server operator tasks.

See also
- List All System Privileges on page 78

SERVER OPERATOR System Privilege
Required to perform server-operator-related tasks.

The SERVER OPERATOR system privilege allows a user to:

- Create databases
- Cache management
- Drop databases
- Start or stop a database
- Start or stop a database engine
- Create, alter, or drop a server
- Create encrypted or decrypted databases
- Create encrypted or decrypted files
- Issue ALTER statement to change transaction logs on a database
- Issue RESTORE statement for a full database restore or to restore the catalog only

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78
**Spatial Objects System Privileges**
System privileges pertaining to performing authorized tasks on spatial objects.

**See also**
- *List All System Privileges* on page 78

**MANAGE ANY SPATIAL OBJECT System Privilege**
Required to manage any spatial objects.

The MANAGE ANY SPATIAL OBJECT system privilege allows a user to issue:

- Issue CREATE, ALTER, or DROP statements on spatial objects
- Issue CREATE, ALTER, or DROP statements on spatial units of measure
- Issue COMMENT statement on spatial units of measure.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Statistics System Privileges**
System privileges pertaining to performing authorized tasks on statistics.

**See also**
- *List All System Privileges* on page 78

**MANAGE ANY STATISTICS System Privilege**
Required to issue CREATE, ALTER, DROP, or UPDATE statements on statistics for any table.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78
Tables System Privileges
System privileges pertaining to performing authorized tasks on tables.

See also
- List All System Privileges on page 78

ALTER ANY TABLE System Privilege
Required to alter any table owned by anyone.

The ALTER DATABASE system privilege allows a user to:
- Issue ALTER or TRUNCATE statement on tables, table partitions, or views owned by any user
- Issue COMMENT statement on tables owned by any user
- Issue COMMENT statement on columns on tables owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

CREATE ANY TABLE System Privilege
Required to create tables owned by any user.

The CREATE ANY TABLE system privilege allows a user to:
- Create tables, including proxy tables, owned by any user
- Issue COMMENT statement on tables owned by any user
- Issue COMMENT statement on columns on tables owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78
CREATE PROXY TABLE System Privilege
Required to create self owned proxy tables.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78

CREATE TABLE System Privilege
Required to create self owned tables.

The CREATE TABLE system privilege allows a user to:
• Create self-owned tables except proxy tables
• Issue COMMENT statement on self-owned tables
• Issue COMMENT statement on columns on self-owned tables

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78

DELETE ANY TABLE System Privilege
Required to delete rows from tables, table partitions, or views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• *GRANT System Privilege Statement* on page 274
• *REVOKE System Privilege Statement* on page 286
• *List All System Privileges* on page 78
DROP ANY TABLE System Privilege
Required to drop tables owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

INSERT ANY TABLE System Privilege
Required to insert rows into tables and views owned by anyone.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

LOAD ANY TABLE System Privilege
Required to execute LOAD command for any table where the -gl server switch is set to DBA.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

SELECT ANY TABLE System Privilege
Required to query tables, views, or materialized views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**TRUNCATE ANY TABLE System Privilege**

Required to execute TRUNCATE command for any table.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**UPDATE ANY TABLE System Privilege**

Required to update rows in tables and views owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also

- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Text Configurations System Privileges**

System privileges pertaining to performing authorized task on text configurations.

See also

- *List All System Privileges* on page 78

**ALTER ANY TEXT CONFIGURATION System Privilege**

Required to alter text configurations owned by any user.

The ALTER ANY TEXT CONFIGURATION system privilege allows a user to:

- Issue ALTER statement on text configurations owned by any user
- Issue COMMENT statement on text configuration owned by any user
Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

CREATE ANY TEXT CONFIGURATION System Privilege
Required to create text configurations owned by other users.

The CREATE ANY TEXT CONFIGURATION system privilege allows a user to:
- Create configurations owned by any user
- Issue COMMENT statement on text configuration owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78

CREATE TEXT CONFIGURATION System Privilege
Required to create self owned text configurations.

The CREATE TEXT CONFIGURATION system privilege allows a user to:
- Create self owned text configurations
- Issue COMMENT statement on self owned text configuration

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- GRANT System Privilege Statement on page 274
- REVOKE System Privilege Statement on page 286
- List All System Privileges on page 78
DROP ANY TEXT CONFIGURATION System Privilege
Required to drop text configurations owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

Triggers System Privileges
System privileges pertaining to performing authorized task on triggers.

See also
• List All System Privileges on page 78

ALTER ANY TRIGGER System Privilege
Required to alter triggers. Users can also issue a COMMENT statement on tables if he or she has the ALTER permission on the table.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

CREATE ANY TRIGGER System Privilege
Required to create triggers. Users can also issue a COMMENT statement on tables if he or she has the ALTER permission on the table.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78
**Users and Login Management System Privileges**
System privileges pertaining to performing authorized task on users and login policies.

**See also**
- *List All System Privileges* on page 78

**CHANGE PASSWORD System Privilege**
Allows users to manage passwords other than their own.

This system privilege can be limited to allow a user to manage passwords for a specific list of users, to manage passwords for any user granted a specific list of roles, or to manage passwords for any existing database user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *Manage Passwords* on page 101
- *GRANT CHANGE PASSWORD Statement* on page 260
- *REVOKE CHANGE PASSWORD Statement* on page 278
- *List All System Privileges* on page 78

**MANAGE ANY LOGIN POLICY System Privilege**
Required to manage login policies.

The MANAGE ANY LOGIN POLICY system privilege allows a user to:
- Issue CREATE, ALTER, or DROP statement on login policies
- Issue COMMENT statement on login policies

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**MANAGE ANY USER System Privilege**
Required to manage users.

The MANAGE ANY USER system privilege allows a user to:
• Issue CREATE, ALTER, or DROP statement on database users (including assigning initial password)
• Define authentication mechanisms for users (Kerberos, Integrated login)
• Issue CREATE or DROP statement on external logins
• Force password change on next login for any user
• Assign a login policy to any user
• Reset the login policy of any user
• Issue COMMENT statement on users, integrated logins, or Kerberos logins.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
• GRANT System Privilege Statement on page 274
• REVOKE System Privilege Statement on page 286
• List All System Privileges on page 78

SET USER System Privilege
Required to allow a user to temporarily assume the specific roles and system privileges (impersonate) of another user.

Suppose a user who is responsible for performing a key task is unavailable. A backup user is identified. At a minimum, this backup user must have sufficient privileges to complete the task; however, depending on the nature of the task to be performed, if the backup user has additional privileges not available to the original user, there is the potential for these additional privileges to result in the task completing differently than for the original user. Negate this potential by allowing the backup user to temporarily assume the roles and system privileges specific to the unavailable user. The backup user "impersonates" the regular user until the key task is finished.

There are two component to the SET USER system privilege. The first component is the SET USER system privilege itself. It is granted by a third party to provide a user with the ability to impersonate another user. The second component is the SETUSER command, which actually impersonate another user. You cannot issue the command to impersonate a user if you have not been granted the privilege to impersonate.

Tip: SET USER is two words when referring to the system privilege, but one word (SETUSER) when referring to the command to actually impersonate another user. You grant the SET USER system privilege, but you issue the SETUSER command to impersonate.

You can limit the granting of the SET USER system privilege to impersonate by allowing users to impersonate:
• Any user in the database
• Any user within a specified list of users
• Any user who is a member of one of the specified roles

For one user to impersonate another user, the grantee (impersonating) user must have been granted at least all of the roles and system privileges, with the same or higher administrative privileges, as those already granted to the target (impersonated) user. The grantee can have been granted additional roles, system privileges, or higher administrative privileges, but not fewer. While a user is impersonating another user, you cannot grant additional privileges to the impersonated user or revoke existing privileges from the impersonating user if doing so invalidate the "at-least" criteria of the SET USER system privilege.

Validation of the at-least criteria occurs when the SETUSER command to impersonate another user is issued, not when the SET USER system privilege is granted to a user. When the SETUSER command is issued, if the grantee fails to meet any of the at-least requirements, a permission denied error message appears.

When one user impersonates another, the user ID of the target user, not the grantee user, is recorded in the audit logs. However, since the act of impersonation (issuance of the SETUSER command) is also recorded in the audit logs, you can determine whether or not a task was executed by the actual user or an impersonating user.

Use the SET USER system privilege only as a temporary measure. While a user is impersonating another user, any roles or system privileges granted to the grantee user are unavailable until the impersonation is terminated. It is strongly recommended that you terminate an impersonation as soon as the required tasks are completed, to allow the grantee to regain their normal roles and system privileges. If you do not deliberately terminate impersonation, it is automatically terminated as soon as the grantee user ends the current session or successfully begins impersonating a different user.

**Scenario 1**
Assume the following:

• There are two users, User1 and User2.
• There are two roles, Role1 and Role2.
• Role1 has been granted the CREATE TABLE system privilege.
• Role2 has been granted the CREATE ANY TABLE system privilege.
• User1 has been granted Role1.
• User2 has been granted Role1 and Role2.

A task requiring the CREATE TABLES system privilege needs to be performed.

The task is usually performed by User1, who is unavailable. User2 has been identified as the backup user to carry out the task. Since both User1 and User2 have been granted Role1, User2 has the required system privilege to perform the task as himself or herself. However, since User2 has also been granted Role2, which includes higher system privileges with respect to creating tables, there is the potential for the task to complete differently than if performed by User1.
To negate this possibility, User2 can impersonate User1 to complete the task.

**Scenario 2 – Meeting At-Least Requirements for Roles**
Assume the following:

- There are two users, User1 and User2.
- There are two roles, Role1 and Role2.
- User1 has been granted Role1.
- User2 has been granted Role1 and Role2.
- User1 has been granted the SET USER system privilege to impersonate User2.
- User2 has been granted the SET USER system privilege to impersonate User1.

User2 can successfully impersonate User1 because they both have been granted Role1, which meets the at-least criteria. However, User1 cannot successfully impersonate User2 because User1 has not been granted Role2 and does not meet the at-least criteria.

**Scenario 3 – Meeting At-Least Requirements for Administrative Options**
Assume the following:

- There are two users, User4 and User5.
- User4 has been granted Role1 with the WITH ADMIN OPTION clause.
- User5 has been granted Role1 with the WITH NO ADMIN OPTION clause.
- User4 has been granted the SET USER system privilege to impersonate User5.
- User5 has been granted the SET USER system privilege to impersonate User4.

Even though both users have been granted Role1, User5 cannot successfully impersonate User4 because he or she has fewer administrative rights to Role1 than User4, which fails the at-least requirement. However, User4 can impersonate User5 because he or she has more administrative rights to Role1 than User5, which meets the at-least requirement.

See also

- *Impersonation* on page 108
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Views System Privileges**
System privileges pertaining to performing authorized tasks on views.

See also

- *List All System Privileges* on page 78
**ALTER ANY VIEW System Privilege**
Required to alter views owned by any user.

The ALTER ANY VIEW system privilege allows a user to:

- Alter views owned by any user
- Issue COMMENT statement on views owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**CREATE ANY VIEW System Privilege**
Required to create views owned by any user.

The CREATE ANY VIEW system privilege allows a user to:

- Create views owned by any user
- Issue COMMENT statement on views owned by any user

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

**See also**
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**CREATE VIEW System Privilege**
Required to create self owned views.

The CREATE VIEW system privilege allows a user to:

- Create self owned views
- Issue COMMENT statement on self owned views

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.
See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**DROP ANY VIEW System Privilege**
Required to drop a view owned by any user.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**Web Services System Privileges**
System privileges pertaining to performing authorized task on Web services.

See also
- *List All System Privileges* on page 78

**MANAGE ANY WEB SERVICE System Privilege**
Required to manage tasks related to Web services.

The MANAGE ANY WEB SERVICE system privilege allows a user to:

- Issue CREATE, ALTER, or DROP statements on Web services
- Issue COMMENT statement on Web services

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default.

See also
- *GRANT System Privilege Statement* on page 274
- *REVOKE System Privilege Statement* on page 286
- *List All System Privileges* on page 78

**List All System Privileges**
List of all system privileges.

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

- ACCESS SERVER LS System Privilege on page 58
- ALTER ANY INDEX System Privilege on page 50
- ALTER ANY MATERIALIZED VIEW System Privilege on page 52
- ALTER ANY OBJECT System Privilege on page 54
- ALTER ANY OBJECT OWNER System Privilege on page 55
- ALTER ANY PROCEDURE System Privilege on page 59
- ALTER ANY SEQUENCE System Privilege on page 64
- ALTER ANY TABLE System Privilege on page 67
- ALTER ANY TEXT CONFIGURATION System Privilege on page 70
- ALTER ANY TRIGGER System Privilege on page 72
- ALTER ANY VIEW System Privilege on page 77
- ALTER DATABASE System Privilege on page 42
- ALTER DATATYPE System Privilege on page 45
- BACKUP DATABASE System Privilege on page 42
- CHANGE PASSWORD System Privilege on page 73
- CHECKPOINT System Privilege on page 42
- COMMENT ANY OBJECT System Privilege on page 55
- CREATE ANY INDEX System Privilege on page 50
- CREATE ANY MATERIALIZED VIEW System Privilege on page 52
- CREATE ANY OBJECT System Privilege on page 55
- CREATE ANY PROCEDURE System Privilege on page 60
- CREATE ANY SEQUENCE System Privilege on page 64
- CREATE ANY TABLE System Privilege on page 67
- CREATE ANY TEXT CONFIGURATION System Privilege on page 71
- CREATE ANY TRIGGER System Privilege on page 72
- CREATE ANY VIEW System Privilege on page 77
- CREATE DATATYPE System Privilege on page 45
- CREATE EXTERNAL REFERENCE System Privilege on page 47
- CREATE MATERIALIZED VIEW System Privilege on page 52
- CREATE MESSAGE System Privilege on page 53
- CREATE PROCEDURE System Privilege on page 60
- CREATE PROXY TABLE System Privilege on page 68
- CREATE TABLE System Privilege on page 68
- CREATE TEXT CONFIGURATION System Privilege on page 71
- CREATE VIEW System Privilege on page 77
- DEBUG ANY PROCEDURE System Privilege on page 46
- DELETE ANY TABLE System Privilege on page 68
- DROP ANY INDEX System Privilege on page 51
Security Management

- DROP ANY MATERIALIZED VIEW System Privilege on page 53
- DROP ANY OBJECT System Privilege on page 56
- DROP ANY PROCEDURE System Privilege on page 61
- DROP ANY SEQUENCE System Privilege on page 64
- DROP ANY TABLE System Privilege on page 69
- DROP ANY TEXT CONFIGURATION System Privilege on page 72
- DROP ANY VIEW System Privilege on page 78
- DROP CONNECTION System Privilege on page 43
- DROP DATATYPE System Privilege on page 45
- DROP MESSAGE System Privilege on page 53
- EXECUTE ANY PROCEDURE System Privilege on page 61
- LOAD ANY TABLE System Privilege on page 69
- INSERT ANY TABLE System Privilege on page 69
- MANAGE ANY DBSPACE System Privilege on page 46
- MANAGE ANY EVENT System Privilege on page 47
- MANAGE ANY EXTERNAL ENVIRONMENT System Privilege on page 48
- MANAGE ANY EXTERNAL OBJECT System Privilege on page 48
- MANAGE ANY LDAP SERVER System Privilege on page 51
- MANAGE ANY LOGIN POLICY System Privilege on page 73
- MANAGE ANY MIRROR SERVER System Privilege on page 58
- MANAGE ANY OBJECT PRIVILEGES System Privilege on page 57
- MANAGE ANY SPATIAL OBJECT System Privilege on page 66
- MANAGE ANY STATISTICS System Privilege on page 66
- MANAGE ANY USER System Privilege on page 73
- MANAGE ANY WEB SERVICE System Privilege on page 78
- MANAGE AUDITING System Privilege on page 61
- MANAGE MULTIPLEX System Privilege on page 59
- MANAGE PROFILING System Privilege on page 43
- MANAGE REPLICATION System Privilege on page 62
- MANAGE ROLES System Privilege on page 63
- MONITOR System Privilege on page 43
- READ CLIENT FILE System Privilege on page 49
- READ FILE System Privilege on page 49
- REORGANIZE ANY OBJECT System Privilege on page 57
- SELECT ANY TABLE System Privilege on page 69
- SERVER OPERATOR System Privilege on page 65
- SET ANY PUBLIC OPTION System Privilege on page 44
- SET ANY SECURITY OPTION System Privilege on page 44
- SET ANY SYSTEM OPTION System Privilege on page 44
Granting a System Privilege to a User

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

Prerequisites

Requires administrative privilege over the system privilege being granted.

Task

Warning! The syntax to grant a system privilege is the same for all system privileges except the CHANGE PASSWORD and SET USER system privileges.

Grant this system privilege using the WITH ADMIN OPTION, WITH NO ADMIN OPTION, or WITH ADMIN ONLY OPTION clause. If you do not specify a clause, the WITH NO ADMIN OPTION clause is used by default. To grant a system privilege to a user, execute one of these statements:

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
</table>
| With full administrative rights            | GRANT system_privilege TO grantee [...]
|                                           | WITH ADMIN OPTION                             |
| With administrative rights only            | GRANT system_privilege TO grantee [...]
|                                           | WITH ADMIN ONLY OPTION                        |
| With no administrative rights              | GRANT system_privilege TO grantee [...]
|                                           | WITH NO ADMIN OPTION                          |

See also
- *GRANT System Privilege Statement* on page 274
- *GRANT CHANGE PASSWORD Statement* on page 260
- *GRANT SET USER Statement* on page 272
**Revoking a System Privilege from a User**
Revoke a specific system privilege and the right to administer the system privilege from specific users.

**Prerequisites**
Requires administrative privilege over the system privilege being revoked.

**Task**

*Warning!* The syntax to revoke a system privilege applies to all system privileges except the CHANGE PASSWORD and SET USER system privileges.

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

<table>
<thead>
<tr>
<th>Administrative Option</th>
<th>Statement</th>
</tr>
</thead>
</table>
| Administrative rights only                               | REVOKE ADMIN OPTION FOR *system_privilege* FROM ...
                                                     |
| System privilege and any administrative rights           | REVOKE *system_privilege* FROM ...

**Example:**
Assuming Mary and Joe were originally granted the BACKUP DATABASE system privilege with administrative rights, execute this statement to remove Mary's administrative rights to the system privilege only, leaving her ability to use the system privilege:

```
REVOKE ADMIN OPTION FOR BACKUP DATABASE FROM Mary
```

Execute this statement to remove the system privilege itself and all administrative rights from Joe:

```
REVOKE BACKUP DATABASE FROM Joe
```

**See also**
- *REVOKE System Privilege Statement* on page 286
- *REVOKE CHANGE PASSWORD Statement* on page 278
- *REVOKE SET USER Statement* on page 284
Users and Privileges Granted System Objects

Information about the current users of a database and their privileges is stored in the database system tables, which are accessible through system views.

Most system tables are owned by the SYS user ID. You cannot log in using to the SYS user ID.

The DBA has SELECT access to all system tables, just as to any other tables in the database. The access of other users to some of the tables is limited. For example, only the DBA has access to the SYS.SYSUSERPERM table, which contains all information about the privileges of users of the database, as well as the passwords of each user ID. However, SYS.SYSUSERPERMS is a view that contains all information in SYS.SYSUSERPERM except passwords, and by default, all users have SELECT access to this view. All privileges and role memberships that are automatically set up in a new database for SYS and PUBLIC system roles, and DBA user can be fully modified.

User ID, Role and Privilege Information in System Tables

System tables containing information about user IDs, roles, and privileges.

All tables and views are owned by the SYS role, and their qualified names are SYS.ISYSUSERPERM, SYS.ISYSTABLEPERM, and so on. Execute the appropriate SELECT queries on these tables generate all the user ID and privilege information stored in the database.

<table>
<thead>
<tr>
<th>Table</th>
<th>Default</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISYSUSERPERM</td>
<td>SELECT ANY TABLE system privilege</td>
<td>Database-level privileges and password for each user ID</td>
</tr>
<tr>
<td>ISYSTABLEPERM</td>
<td>PUBLIC</td>
<td>All privileges on table given by the GRANT commands</td>
</tr>
<tr>
<td>ISYSCOLPERM</td>
<td>PUBLIC</td>
<td>All columns with UPDATE privilege given by the GRANT command</td>
</tr>
<tr>
<td>ISYSPROCPERM</td>
<td>PUBLIC</td>
<td>Each row holds one user granted privilege to use one procedure</td>
</tr>
</tbody>
</table>

User ID, Role, and Privilege Information in System Views

System views containing information about user IDs, roles, and privileges.

In addition to this list, there are tables and views containing information about each object in the database.
### View Definitions

<table>
<thead>
<tr>
<th>View</th>
<th>Default</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSUSERAUTH</td>
<td>SELECT ANY TABLE system privilege</td>
<td>All information in SYSUSERPERM except user numbers</td>
</tr>
<tr>
<td>SYSUSERPERMS</td>
<td>PUBLIC</td>
<td>All information in SYSUSERPERM except passwords</td>
</tr>
<tr>
<td>SYSUSERLIST</td>
<td>PUBLIC</td>
<td>All information in SYSUSERAUTH except passwords</td>
</tr>
<tr>
<td>SYSTABAUTH</td>
<td>PUBLIC</td>
<td>Information from SYSTABLEPERM in a more readable format</td>
</tr>
<tr>
<td>SYSCOLAUTH</td>
<td>PUBLIC</td>
<td>Information from SYSCOLPERM in a more readable format</td>
</tr>
<tr>
<td>SYSPROCAUTH</td>
<td>PUBLIC</td>
<td>Information from SYSPROCPERM in a more readable format</td>
</tr>
</tbody>
</table>

**Stored Procedure to Map System Privileges to System Roles**

The `sp_sys_priv_role_info` stored procedure generates a report that maps each system privilege role to a system role.

A separate row is generated for each system privilege. No system privileges are required to execute the procedure.

**Object-Level Privileges**

Database object-level privileges can be granted to and revoked from users.

**Ownership Privileges of Database Objects**

Ownership of a database object carries with it privileges to carry out actions on that object. The creator of a database object may or may not be the owner of that object.

The creator of a database object may not necessarily be its owner. 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 may modify the structure of that table, for instance, or may grant privileges to other database users to update the information within the table.

**Note:** The owner of a table can load data if he or she has sufficient privilege or the server was started with the `-gl all` switch on the command line or configuration file. Ownership or the CREATE ANY OBJECT system privilege are insufficient to issue the LOAD TABLE command. The INSERT privilege on the table is also required.

A user with the ALTER ANY OBJECT system privilege can modify any database object (regardless of owner) that could be created using the CREATE ANY OBJECT system.
privilege. A user with the CREATE ANY OBJECT system privilege can create database objects to be owned by other users.

**Inheritance of Database Privileges**

You can grant database privileges directly to users or it can be inherited through role membership.

<table>
<thead>
<tr>
<th>Privilege Name</th>
<th>Supported By Database Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL</td>
<td>Tables, Views, Materialized Views</td>
<td>Allows a user to perform all tasks associated with tables, views and materialized views.</td>
</tr>
<tr>
<td>ALTER</td>
<td>Tables</td>
<td>Allows a user to alter the structure of a table.</td>
</tr>
<tr>
<td>CREATE</td>
<td>Dbspaces</td>
<td>Allows a user to create on the dbspace. The additional privileges required depend on the object that is being created. For example, to create a table, one of CREATE TABLE, CREATE ANY TABLE, or CREATE ANY OBJECT is required.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Tables, View</td>
<td>Allows a user to delete rows from the table or view.</td>
</tr>
<tr>
<td>EXECUTE</td>
<td>Procedure, User-defined Functions</td>
<td>Allows a user to execute the procedure or user-defined function.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Table, Views</td>
<td>Allows a user to insert rows into the table or view.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Tables</td>
<td>Allows a user to load the table if the -gl database option is set to anything other than NONE.</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>Tables</td>
<td>Create indexes on a table, and to create foreign keys that reference a table</td>
</tr>
<tr>
<td>SELECT</td>
<td>Table, Views</td>
<td>Look at information in a table or view</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>Table, Materialized Views</td>
<td>Allows a user to truncate the table or materialized view.</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Tables, Views</td>
<td>Update rows in a table or view.</td>
</tr>
<tr>
<td>USAGE</td>
<td>Sequence Generators</td>
<td>Allows a user to evaluate the current or next value in the sequence.</td>
</tr>
</tbody>
</table>

In a multiplex, only write servers can modify table privileges on tables owned by the write server.
Grant and Revoke Object-Level Privileges

Users can be granted or revoked combinations of privileges to define their access to a database objects

Granting the ALTER Privilege on Tables

Grant the privilege to alter the structure of a table. This privilege does not apply to views.

Prerequisites

Requires one of:

- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the ALTER object privilege on the table with the WITH GRANT OPTION clause.
- You own the table.

Task

To grant the ALTER privilege, enter:

```
GRANT ALTER
ON table_name
TO userID [,..]
```

See also

- *GRANT Object-Level Privilege Statement* on page 265
- *Granting the Right to Administer an Object-Level Privilege* on page 90

Granting the DELETE Privilege on Tables and Views

Grant the privilege to delete all data in a specified table or view.

Prerequisites

Requires one of:

- The MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the DELETE object privilege on the table with the WITH GRANT OPTION clause.
- You own the table.

Task

To grant the DELETE privilege, enter:

```
GRANT DELETE
ON table_name
TO userID [,..]
```
See also

• *GRANT Object-Level Privilege Statement* on page 265
• *Granting the Right to Administer an Object-Level Privilege* on page 90

**Granting the INSERT Privilege on Tables and Views**
Grant the privilege to insert data into a table or view.

**Prerequisites**
Requires one of:

- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the INSERT object privilege on the table with the WITH GRANT OPTION clause.
- You own the table.

**Task**
To grant the INSERT privilege, enter:

```
GRANT INSERT
  ON table_name
  TO userID [, ...]
```

See also

• *GRANT Object-Level Privilege Statement* on page 265
• *Granting the Right to Administer an Object-Level Privilege* on page 90

**Granting the LOAD Privilege on Tables**
Grant the privilege to load a specified table.

**Prerequisites**
Requires one of:

- MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the LOAD object privilege with the WITH GRANT OPTION clause on the table.
- You own the table.

**Task**
To grant the LOAD privilege, enter:

```
GRANT LOAD
  ON table_name
  TO userID [, ...]
```
See also

- *GRANT Object-Level Privilege Statement* on page 265
- *Granting the Right to Administer an Object-Level Privilege* on page 90

**Granting the REFERENCES Privilege on Tables**

Grant the privilege to indexes and to foreign keys on a table. This privilege does not apply to views. This privilege can be restricted to a set of columns in the table.

**Prerequisites**

Requires one of:

- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the REFERENCE object privilege on the table with the WITH GRANT OPTION clause.
- You own the table.

**Task**

To grant the REFERENCES privilege, enter:

```
GRANT REFERENCES column_name
    ON table_name
    TO userID [, ...]
```

**Example:**

This statement grants the REFERENCES privilege to user Joe on columns Col_1 and Col_2 in the table named sales_table:

```
GRANT REFERENCES Col_1, Col_2 ON sales_table
    TO Joe
```

See also

- *GRANT Object-Level Privilege Statement* on page 265
- *Granting the Right to Administer an Object-Level Privilege* on page 90

**Granting the SELECT Privilege on Tables and Views**

Grant the privilege to select data in a table or view, but not to alter it. This privilege can be restricted to a set of columns in the table.

**Prerequisites**

Requires one of:

- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the SELECT object privilege on the table with the WITH GRANT OPTION clause.
• You own the table.

Task
To grant the SELECT privilege, enter:

```
GRANT SELECT column_name
    ON table_name
    TO userID [, ...]
```

Example:
This statement grants the SELECT privilege to user Joe on columns Col_1 and Col_2 in the table named sales_table:

```
GRANT SELECT Col_1, Col_2 ON sales_table
    TO Joe
```

See also
• GRANT Object-Level Privilege Statement on page 265
• Granting the Right to Administer an Object-Level Privilege on page 90

Granting the TRUNCATE Privilege on Tables
Grant the privilege to truncate a specified table.

Prerequisites
Requires one of:

• The MANAGE ANY OBJECT PRIVILEGE system privilege.
• You have been granted the TRUNCATE object privilege with the WITH GRANT OPTION clause on the table.
• You own the table.

Task
To grant the TRUNCATE privilege, enter:

```
GRANT TRUNCATE
    ON table_name
    TO userID [, ...]
```

See also
• GRANT Object-Level Privilege Statement on page 265
• Granting the Right to Administer an Object-Level Privilege on page 90
Granting the UPDATE Privilege on Tables and Views
Grant the privilege to modify the data in a table or view. This privilege can be restricted to a set of columns in the table.

Prerequisites
Requires one of:

- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.
- You have been granted the UPDATE object privilege on the table with the WITH GRANT OPTION clause.
- You own the table.

Task
To grant the UPDATE privilege, enter:

```
GRANT UPDATE column_name ON table_name TO userID [,,...]
```

Example:
This statement grants the UPDATE privilege to user Joe on columns Col_1 and Col_2 in the table named sales_table:

```
GRANT UPDATE Col_1, Col_2 ON sales_table TO Joe
```

See also
- `GRANT Object-Level Privilege Statement` on page 265
- `Granting the Right to Administer an Object-Level Privilege` on page 90

Granting the Right to Administer an Object-Level Privilege
Grant the privilege to allow a user to pass a specific object privilege on to other users.

Prerequisites
At least one of these conditions:

- You created the table.
- You have been granted privileges on the table with the ADMIN OPTION.
- You have been granted LOAD and TRUNCATE object privileges.
- You have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege. If the LOAD or TRUNCATE object privilege is granted using the WITH GRANT OPTION clause, the grantee can then grant the object privilege to other users, but is limited to those
tables specified in the original GRANT statement. Under this scenario, the grantee does
not need the MANAGE ANY OBJECT PRIVILEGE system privilege.

Task

1. Connect to the database.
2. To grant the right to grant a privilege to another user, enter:

   \[
   \text{GRANT } \text{Object_privilege\_name} \\
   \text{ON } \text{table\_name} \\
   \text{TO } \text{userID} \,[,...] \\
   \text{WITH GRANT OPTION}
   \]

   Example:

   This statement grants the privilege to Mary to perform deletes on the table Sales:
   \[
   \text{GRANT DELETE ON Sales TO Mary}
   \]

   This statement grants the right to Joe to both perform deletes on the table Sales and grant
   the DELETE privilege to other users:
   \[
   \text{GRANT DELETE ON Sales TO Joe} \\
   \text{WITH GRANT OPTION}
   \]

See also

- GRANT Object-Level Privilege Statement on page 265
- Granting the Right to Administer an Object-Level Privilege on page 90

Granting the CREATE Privilege on Dbspaces
Grant the privilege to create database objects in the specified dbspace.

Prerequisites
Requires the MANAGE ANY DBSPACE system privilege.

Task
To grant the CREATE privilege, enter:

\[
\text{GRANT CREATE} \\
\text{ON } \text{dbspace\_name} \\
\text{TO } \text{userID} \,[,...]
\]

See also

- GRANT CREATE Statement on page 264
Granting the EXECUTE Privilege on Functions and Procedures

Grant the privilege to run a procedure or user-defined function.

**Prerequisites**
Requires one of:

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

**Task**
To grant the EXECUTE privilege, enter:

```
GRANT EXECUTE
ON procedure_name
TO userID [, ...]
```

**See also**
- *GRANT EXECUTE Statement on page 265*

Granting the USAGE Privilege on Sequence Generators

Grant the privilege to evaluate the current or next value in a sequence.

**Prerequisites**
Requires one of:

- The MANAGE ANY OBJECT PRIVILEGE system privilege.
- You own the sequence generator.

**Task**
To grant the USAGE privilege, enter:

```
GRANT USAGE
ON sequence_name
TO userID [, ...]
```

**See also**
- *GRANT USAGE ON SEQUENCE Statement on page 277*

Revoking an Object-Level Privilege

Remove the ability of a user to use a specific object-level privilege or grant the privilege to other users.

**Prerequisites**
Grantor must have at least one of these conditions:
Be the original grantor of the privilege that is being revoked
Have the MANAGE ANY OBJECT PRIVILEGE system privilege

Task
If you revoke a privilege from a user who has been granted a privilege with the WITH GRANT OPTION clause, then everyone who that user in turn granted the privilege to also has their privilege revoked. For example, you granted UserA the SELECT privilege with the WITH GRANT OPTION clause. UserA then grants the SELECT privilege to UserB. If you revoke the SELECT privilege from UserA, it is also revoked for UserB.

The REVOKE command applies to the object-level privilege itself, not to any administrative right granted on the privilege. Therefore, you cannot revoke administrative rights only and leave the object-level privilege intact. To correctly remove a user's administrative rights only to an object-level privilege, you must first revoke the privilege and then re-grant the privilege without the WITH GRANT OPTION clause.

1. To revoke an object-level privilege, including any administrative privilege, execute:

   REVOKE  object_privilege_name
   ON  table_name
   FROM  userID [,...

2. (Optional) To then re-grant the object-level privilege without administrative rights, execute:

   GRANT  object_privilege_name
   ON  table_name
   TO  userID [,...

Example:
This example assumes that Joe was granted the right to both perform deletes on the table Sales, and grant the DELETE privilege on the table to other users.

This statement revokes all DELETE privileges on the table Sales, which by definition includes any administrative rights:

   REVOKE DELETE ON Sales FROM Joe

This statement re-grants the privilege only, with no administrative rights:

   GRANT DELETE ON Sales TO Joe

See also
• REVOKE Object-Level Privilege Statement on page 281
• REVOKE CREATE Statement on page 280
• REVOKE EXECUTE Statement on page 281
• REVOKE USAGE ON SEQUENCE Statement on page 289
**Privileges Required to Manage Table Objects in a Dbspace**

There are specific system privileges required to create or move a table object in a dbspace.

Requires the CREATE privilege on the dbspace. The CREATE privilege in a dbspace can be granted to or revoked from a user or a role. Any member in a role inherits CREATE privilege from the role. By default, CREATE privilege on IQ_SYSTEM_MAIN, IQ_SYSTEM_TEMP, and SYSTEM is granted to PUBLIC. For other IQ main dbspaces, the system administrator must explicitly grant CREATE privilege on the dbspace before a role or user can create or move objects into that dbspace. For example, if a table is to be placed on a new IQ main dbspace, the user must have CREATE privilege on that dbspace. Users must also have CREATE ANY OBJECT privilege to create objects.

**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 he or she is 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 still need 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 he or she is 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.

See also
- `-gl iqsrv16 Server Option` on page 303
- `-gu iqsrv16 database server option` on page 304
- `-gk iqsrv16 database server option` on page 302

**Revoking the Privilege to Run a Procedure**
Remove the privilege to execute or call a specific procedure.

**Prerequisites**
Revoker must either:

- Be the original grantor of the privilege that is being revoked
- Have the MANAGE ANY OBJECT PRIVILEGE system privilege

**Task**
To revoke the EXECUTE privilege to run a specific procedure, execute:

```
REVOKE EXECUTE ON procedure_name
FROM grantee [, ...]
```

See also
- *REVOKE EXECUTE Statement* on page 281
**How User Privilege Conflicts Are Resolved**

Roles introduce complexities in the granting of privileges of individual users.

Suppose user Joe has been individually granted SELECT and UPDATE privileges on a specific table. Joe is also a member of two roles, one of which has no access to the table at all, and one of which has only SELECT access. What are the privileges in effect for Joe?

This is how SAP Sybase IQ determines whether a user ID has privilege to carry out a specific action:

1. If the user ID has DBA privileges, he or she can carry out any action in the database. If the user has specific system privileges granted, he or she can have the privileges to carry out only those authorized tasks associated with the system privileges.
2. Otherwise, privilege depends on the privileges assigned to the individual user. If the user ID has been granted privilege to carry out the action, the action is allowed to proceed.
3. If no individual settings have been made for that user, privilege depends on the privileges of each of the roles of which the user is a member. If any of these roles has privilege to carry out the action, the user ID has privilege by virtue of membership in that role, and the action is allowed to proceed.

If you do not want a specific user to access a particular table, view, or procedure, do not make that user a member of a role that has privileges on that object.

This approach minimizes problems associated with the order in which privileges are set.

**Stored Procedure to Display Object-Level Privileges Granted**

Execute the `sp_objectpermission` stored procedure to generate a report on object-level privileges granted to the specified role or user name or object privileges granted on the specified object or dbspace.

The report includes the user ID of the privilege grantor and grantee, the object name and owner, the privilege granted and whether the grantee can in turn grant the privilege to other users.

No system privileges are required to execute the procedure on your user ID. To execute `sp_objectpermission` on other users or a dbspace, you must have MANAGE ANY OBJECT PRIVILEGE or MANAGE ANY DBSPACE privilege, respectively.

**See also**
- `sp_objectpermission System Procedure` on page 362

**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 to SAP Sybase IQ privileged system procedures only, 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 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 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 privilege. However, since the PUBLIC role is not the owner of the system procedures, and is not granted any system privileges, system privileges required to run any underlying authorized tasks must be granted directly or indirectly to the user.

By default, a new 16.0 database runs all privileged system procedures using the SYSTEM PROCEDURE INVOKER model. By default, a pre-16.0 upgraded database runs privileged system procedures using a combination of both the SYSTEM PROCEDURE DEFINER and SYSTEM PROCEDURE INVOKER models. In the combined model, all pre-16.0 privileged system procedures run using the SYSTEM PROCEDURE DEFINER model, and any privileged system procedures introduced with 16.0 (or any future release) run using the SYSTEM PROCEDURE INVOKER model. The default security model can be overridden during database creation or upgrade, or changed any time thereafter. However, this is not recommended as it may result in loss of functionality on custom stored procedures and applications.

**Grant the Ability to Run a Privileged System Procedure**

The process by which you grant the ability to run a privileged system procedure is dependant on the security model under which it runs.

For a privileged system procedure using the SYSTEM PROCEDURE DEFINER model, grant EXECUTE privilege on the system procedure to the user:

```
GRANT EXECUTE ON sys_procedure_name TO grantee [,....]
```

For a privileged system procedure using the SYSTEM PROCEDURE INVOKER model, grant the underlying system privileges required by the system procedure to the user. Use `sp_proc_priv()` to identify the system privileges required to run a system procedure.

```
GRANT system_privilege_name TO grantee [,....]
```

**See also**

- *GRANT EXECUTE Statement* on page 265
Revoke the Ability to Run a Privileged System Procedure

The process by which you revoke the ability to run a privileged system procedure is dependant on the security model under which it runs.

For a privileged system procedure using the SYSTEM PROCEDURE DEFINER model, revoke the EXECUTE privilege on the system procedure from the user:

```
REVOKE EXECUTE ON sys_procedure_name
    FROM grantee [,...]
```

For a privileged system procedure using the SYSTEM PROCEDURE INVOKER model, revoke the underlying system privileges required by the system procedure from the user:

```
REVOKE system_privilege_name
    FROM grantee [,...]
```

See also

- **REVOKE EXECUTE Statement** on page 281

Determine Security Model Used by a Database

By default, a new 16.0 database runs privileged system procedures using the SYSTEM PROCEDURE INVOKER model only, while a pre-16.0 upgraded database runs privileged system procedures using a combination of both the SYSTEM PROCEDURE DEFINER and SYSTEM PROCEDURE INVOKER models.

However, it is possible to override the defaults during database creation or upgrade. To verify the security model of a database, execute:

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

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

Remember, 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.

A new or upgraded 16.0 database cannot be configured to run all system procedures using the SYSTEM PROCEDURE DEFINER model.

Pre-16.0 Privileged System Procedures

A list of pre-16.0 privileged system procedures.

**Pre-16.0 privileged system procedures that use the combined security model**

For these privileged system procedures, if the database is configured to run using SYSTEM PROCEDURE DEFINER, you only need EXECUTE privilege on the procedure to run it. If
the database is configured to run using SYSTEM PROCEDURE INVOKER, you need the
individual privileges that each procedure requires to run successfully. Refer to the
documentation for each procedure's required system privileges.
<table>
<thead>
<tr>
<th>Function</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa_audit_string</td>
<td>sa_text_index_vocab</td>
<td>sp_iqmodifylogin</td>
</tr>
<tr>
<td>sa_checkpoint_execute</td>
<td>sa_text_index_vocab_nchar</td>
<td>sp_iqmpxcheckdqpconfig</td>
</tr>
<tr>
<td>sa_clean_database</td>
<td>sa_unload_cost_model</td>
<td>sp_iqmpxdumptvlog</td>
</tr>
<tr>
<td>sa_column_stats</td>
<td>sa_user_defined_counter_add</td>
<td>sp_iqmpxfilestatus</td>
</tr>
<tr>
<td>sa_conn_activity</td>
<td>sa_user_defined_counter_set</td>
<td>sp_iqmpxinconnpoolinfo</td>
</tr>
<tr>
<td>sa_conn_compression_info</td>
<td>sa_validate</td>
<td>sp_iqmpxincheartbeatinfo</td>
</tr>
<tr>
<td>sa_conn_info</td>
<td>sp_iq_reset_identity</td>
<td>sp_iqmpxinfo</td>
</tr>
<tr>
<td>sa_conn_list</td>
<td>sp_iqaddlogin</td>
<td>sp_iqmpxversioninfo</td>
</tr>
<tr>
<td>sa_conn_options</td>
<td>sp_iqbackupdetails</td>
<td>sp_iqobjectinfo</td>
</tr>
<tr>
<td>sa_conn_properties</td>
<td>sp_iqcardinality_analysis</td>
<td>sp_iqkeys</td>
</tr>
<tr>
<td>sa_db_info</td>
<td>sp_iqclient_lookup</td>
<td>sp_iqprocedure</td>
</tr>
<tr>
<td>sa_db_list</td>
<td>sp_iqcolumn</td>
<td>sp_iqprocparm</td>
</tr>
<tr>
<td>sa_db_properties</td>
<td>sp_iqconnection</td>
<td>sp_iqrebuildindex</td>
</tr>
<tr>
<td>sa_disable_auditing_type</td>
<td>sp_iqconstraint</td>
<td>sp_iqrename</td>
</tr>
<tr>
<td>sa_disk_free_space</td>
<td>sp_iqcontext</td>
<td>sp_iqrestorationaction</td>
</tr>
<tr>
<td>sa_enable_auditing_type</td>
<td>sp_iqcopyloginpolicy</td>
<td>sp_iqrowdensity</td>
</tr>
<tr>
<td>sa_external_library_unload</td>
<td>sp_iqcursorinfo</td>
<td>sp_iqsetcompression</td>
</tr>
<tr>
<td>sa_flush_cache</td>
<td>sp_iqdatastype</td>
<td>sp_iqsharedtempdistrib</td>
</tr>
<tr>
<td>sa_flush_statistics</td>
<td>sp_iqdbsize</td>
<td>sp_iqshowcompression</td>
</tr>
<tr>
<td>sa_get_histogram</td>
<td>sp_iqdbspace</td>
<td>sp_iqshowpsexec</td>
</tr>
<tr>
<td>sa_get_request_profile</td>
<td>sp_iqdbspaceinfo</td>
<td>sp_iqspaceinfo</td>
</tr>
<tr>
<td>sa_get_request_times</td>
<td>sp_iqdbspaceinfo</td>
<td>sp_iqspaceused</td>
</tr>
<tr>
<td>sa_get_table_definition</td>
<td>sp_iqconnection</td>
<td>sp_iqstatistics</td>
</tr>
<tr>
<td>sa_get_user_status</td>
<td>sp_iqcontext</td>
<td>sp_iqsysmon</td>
</tr>
<tr>
<td>sa_index_density</td>
<td>sp_iqcopyloginpolicy</td>
<td>sp_iqtabelle</td>
</tr>
<tr>
<td>sa_index_levels</td>
<td>sp_iqrdblogfabricpolicy</td>
<td>sp_iqtablesize</td>
</tr>
<tr>
<td>sa_install_feature</td>
<td>sp_iqspaceinfo</td>
<td>sp_iqtableuse</td>
</tr>
<tr>
<td>sa_java_loaded_classes</td>
<td>sp_iqstatus</td>
<td>sp_iqtablesizie</td>
</tr>
<tr>
<td>sa_list_external_library</td>
<td>sp_iqtable</td>
<td>sp_iqtableuse</td>
</tr>
<tr>
<td>sa_load_cost_model</td>
<td>sp_iqtransaction</td>
<td>sp_iqtransaction</td>
</tr>
<tr>
<td>sa_materialized_view_can_be_immediate</td>
<td>sp_iqunusedcolumn</td>
<td>sp_iqunusedcolumn</td>
</tr>
<tr>
<td>sa_procedure_profile</td>
<td>sp_iqunusedindex</td>
<td>sp_iqunusedindex</td>
</tr>
<tr>
<td>sa_procedure_profile_summary</td>
<td>sp_iqunusedtable</td>
<td>sp_iqversionuse</td>
</tr>
<tr>
<td>sa_recompile_views</td>
<td>sp_iqview</td>
<td>sp_iqwho</td>
</tr>
<tr>
<td>sa_refresh_materialized_views</td>
<td>sp_iqworkmon</td>
<td>st_geometry_load_shapefile</td>
</tr>
<tr>
<td>sa_refresh_text_indexes</td>
<td></td>
<td>xp_cmdshell</td>
</tr>
<tr>
<td>sa_remove_index_consultant_analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Pre-16.0 privileged system procedures that run with invoker privileges regardless of the security model

These pre-16.0 privileged system procedures run with the privileges of the user running the procedure, not the owner of the procedure, regardless of the security model setting. This means that in addition to requiring EXECUTE privilege on the system procedure, the user must be granted additional system privileges required by the system procedure. Refer to the documentation for the required system privileges.

- `sa_describe_shapefile`
- `sa_get_user_status`
- `sa_locks`
- `sa_performance_diagnostics`
- `sa_report_deadlocks`
- `sa_text_index_stats`

### Manage Passwords

A user can be granted the ability to manage the password of other users. Password management can be configured to require one or two users to complete a password change.

#### 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) is 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.

**Task**
A user can be granted the ability to change the password of any user in the database (ANY) or 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 only be granted when using the ANY clause.

If no clause is specified, ANY, WITH NO ADMIN OPTION is used by default.

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 the CHANGE PASSWORD system privilege is regranted to a user with lesser 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.

To grant the CHANGE PASSWORD system privilege, execute one of these statements:

<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any database user, with full administrative rights</td>
<td>GRANT CHANGE PASSWORD (ANY) TO user_ID WITH ADMIN OPTION</td>
</tr>
<tr>
<td>Any database user, with administrative rights only</td>
<td>GRANT CHANGE PASSWORD (ANY) TO user_ID WITH ADMIN ONLY OPTION</td>
</tr>
<tr>
<td>Any database user, with no administrative rights</td>
<td>GRANT CHANGE PASSWORD (ANY) TO user_ID WITH NO ADMIN OPTION</td>
</tr>
<tr>
<td>Specified users, with no administrative rights</td>
<td>GRANT CHANGE PASSWORD (target_users_list) TO user_ID WITH NO ADMIN OPTION</td>
</tr>
<tr>
<td>Grant Type</td>
<td>Statement</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Any member of specified roles, with no administrative rights</td>
<td>GRANT CHANGE PASSWORD (ANY WITH ROLES target_roles_list) TO user_ID WITH NO ADMIN OPTION</td>
</tr>
<tr>
<td>Specified users, or any member of specified roles, with no administrative rights</td>
<td>GRANT CHANGE PASSWORD (target_users_list), (ANY WITH ROLES target_roles_list) TO user_ID WITH NO ADMIN OPTION</td>
</tr>
</tbody>
</table>

**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
```

**See also**

- *GRANT CHANGE PASSWORD Statement* on page 260
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.

Task
The CHANGE PASSWORD system privilege can be granted to a user multiple times, using different clauses. For example, UserA 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 form of the clause used for the GRANT must be used to revoke it. Continuing with the example, if the system privilege is revoked from UserA using the ANY clause, the grant with the target_users_list clause remains in effect. The net effect is that UserA is now limited to managing the passwords of users on the target_users_list. Alternately, if the system privilege is revoked from UserA using the target_users_list clause, the grant with the ANY clause remains in effect. The net effect in this scenario is that UserA can continue to manage the passwords of any user in the database.

To revoke the CHANGE PASSWORD system privilege, execute one of these statements:

<table>
<thead>
<tr>
<th>Revoke Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative rights to system privilege only</td>
<td>REVOKE ADMIN OPTION FOR CHANGE PASSWORD ( ANY )</td>
</tr>
<tr>
<td></td>
<td>FROM user_ID [,...]</td>
</tr>
<tr>
<td>System privilege to manage password of any database user, including administrative rights</td>
<td>REVOKE CHANGE PASSWORD</td>
</tr>
<tr>
<td></td>
<td>FROM user_ID [,...]</td>
</tr>
<tr>
<td>System privilege to manage password of specified users</td>
<td>REVOKE CHANGE PASSWORD ( target_users_list )</td>
</tr>
<tr>
<td></td>
<td>FROM user_ID [,...]</td>
</tr>
<tr>
<td>System privilege to manage password of specified roles</td>
<td>REVOKE CHANGE PASSWORD ( ANY WITH ROLES target_roles_list )</td>
</tr>
<tr>
<td></td>
<td>FROM user_ID [,...]</td>
</tr>
</tbody>
</table>

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 the ability 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, Marketing2) FROM Joan

See also
• REVOKE CHANGE PASSWORD Statement on page 278

Changing a Password – Single Control
A single user can manage the password of another user.

Prerequisites
• Requires the CHANGE PASSWORD system privilege.
• The managing user has been granted the right to change the password of the target user.

Task
At a command prompt, type:
ALTER USER userID IDENTIFIED BY password

See also
• Case-sensitivity of User IDs and Passwords on page 122
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 cancelled 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 on. 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, type 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.

See also

- Case-sensitivity of User IDs and Passwords on page 122
- ALTER USER Statement on page 242
- GRANT CONNECT Statement on page 262
- sp_iqpassword Procedure on page 361

Enabling Dual Control for Changing Passwords

Require input from two administration users to change the password of another user.

Prerequisites

Requires the MANAGE ANY LOGIN POLICY OPTION system privilege.
Task
Dual control for managing passwords is a configurable option of a login policy. By default, it is disabled (OFF).
To enable the option, execute:

```
ALTER LOGIN POLICY policy-name
CHANGE_PASSWORD_DUAL_CONTROL=ON
```

See also
- `ALTER LOGIN POLICY Statement` on page 233
- `CREATE LOGIN POLICY Statement` on page 248

Changing a Password – Dual Control
Two users are required to manage the password of another user.

Prerequisites
- Requires 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.

Task
1. At a command prompt, the first managing user types:
   ```
   ALTER USER userID IDENTIFIED FIRST BY password_part1
   ```
2. At a command prompt, the second managing user types:
   ```
   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
  ```

See also
- `Case-sensitivity of User IDs and Passwords` on page 122
Impersonation

A user can temporarily assume the roles and system privileges of another user (also known as impersonation) to perform operations, provided he or she already has the minimum required privileges to perform the task to begin with.

For example, suppose User1 is responsible for performing a key task, but he or she is unavailable. User2 has sufficient privileges to complete the task, but has additional privileges not available to User1. If User2 performs the task, it may complete differently than when performed by User1. To avoid this, User2 temporarily assumes the roles and system privileges specific to User1, and performs the task.

Impersonation is achieved by first granting a user the SET USER system privilege, and then issuing the SETUSER statement to initiate the impersonation.

**Note:** The SET USER system privilege is two words; the SETUSER statement is one word.

When you grant the SET USER system privilege, you can define the scope of impersonation as:

- 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*).

To impersonate another user, you must have been granted, at minimum, all of the roles and system privileges with the same or higher administrative privileges, as the user you are impersonating. This is called the *at-least* criteria.

Impersonation criteria validation occurs when the SETUSER statement is executed, not when the SET USER system privilege is granted. This is because roles and system privileges granted to both the impersonator and impersonatee may change over time. If at the moment of SETUSER execution, if the user does not meet all criteria, impersonation does not begin. However, if the all criteria is met on a subsequent SETUSER execution, impersonation begins.

You may ask why, if a user already has all the privileges he or she needs to perform a task that someone else normally performs, the user does not just perform the task as themselves. The reason is that if the impersonating user has more privileges than he or she needs to perform the task, even though the extra privileges are not required for the task, the additional privileges can affect the output of the task. By impersonating the user who normally performs the task, it negates this possibility.

For example, assume the following conditions:

- There are two users, User1 and User2.
- There are two roles, Role1 and Role2.
• Role1 has been granted the CREATE TABLE system privilege.
• Role2 has been granted the CREATE ANY TABLE system privilege.
• User1 has been granted Role1.
• User2 has been granted Role1 and Role2.

A task requiring the CREATE TABLE system privilege must be performed.

The task is usually performed by User1, who is unavailable. User2 has been identified as the backup user to carry out the task.

Since both User1 and User2 have been granted Role1, User2 has the required system privilege to perform the task. However, since User2 has also been granted Role2, which includes higher system privileges with respect to creating tables (the ability to create tables owned by other users), there is the potential for the task to complete differently than if it was performed by User1.

To negate this possibility, instead of User2 running the task, User2 impersonates User1 and completes the task.

Once the SET USER system privilege to impersonate another user is granted, it remains in effect until it is revoked.

Once you issue the SETUSER statement, and successful impersonation begins, it remains in effect until you manually terminated the impersonation, begin impersonating another user, or the current session ends. It is recommended that impersonation be terminated as soon as the required tasks are completed.

While a user is impersonating another user, roles and privileges and their related administrative rights can be granted to or revoked from the impersonator or impersonatee as long as doing so does not violate the criterion behind the impersonation. If the grant or revoke violates the impersonation criteria, an error message appears, and the statement fails.

For example, UserA is successfully impersonating UserB. Someone tries to grant a new role to UserA, but not to UserB. Since this grant would not result in a violation of the criteria for UserA to impersonate UserB (UserA still has at least all of the roles and privileges granted to UserB), the grant is successful. If, however, the new role grant was to UserB instead of UserA, the grant statement would fail because it would result in a violation of the criteria (UserB would have been granted more roles than UserA).

In a Multiplex configuration, if an impersonation is active in a connection present in the coordinator, and an attempt is made to grant or revoke roles and privileges that would violate the impersonation criterion, the connection containing the active impersonation is dropped. Since dropping the connection also terminates the impersonation, violation of criteria is no longer an issue, the GRANT or REVOKE statement executes successfully.

When you impersonate another user, the user ID of the impersonated user appears in the transaction log, not yours. However, since the SETUSER statement also appears in the transaction log, it is easy to determine whether the task was executed by the actual user or by someone using impersonation.
Understand the 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 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. For example, User1 is granted Role1 with the WITH ADMIN OPTION clause, User2 is granted Role1 with the WITH ADMIN ONLY clause, and User3 is granted Role1 with the WITH NO ADMIN OPTION clause. User1 and User2 are said to be granted Role1 with similar administrative rights. User1 and User2 are also said to be granted Role1 with higher administrative rights than User3.

4. If the target user has been granted a system privilege which supports extensions, the clauses used to grant the system privilege to the impersonator are a super-set of those used for the target user. Currently, only the SET USER and CHANGE PASSWORD system privileges support extensions.

**Note:**
- 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, Role1, Role2, respectively, while the target list grants of the target user contain User1, 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 themselves. Therefore, if the target user has been 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 the second and third criterion is met, consider the following scenario:

- There are five users: User1, User2, User3, User4, and User5.
- There are two roles: Role1 and Role2.
- User1 was granted the SET USER system privilege with the ANY clause.
- User2 was granted the SET USER system privilege with the target_users_list clause for User1 and User4.
- User3 was 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 was granted the SET USER system privilege with the ANY clause and the target_roles_list clause for Role1.
- User5 was 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)
Security Management

- 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 the first and fourth criteria 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)

Validation of criterion 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.

**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) is 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.

**Task**
A user can be granted the ability to impersonate any user in the database (ANY) or 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 only be granted when using the ANY clause.

If no clause is specified, ANY is used by 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 granted.

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

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

<table>
<thead>
<tr>
<th>Grant Type</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>System privilege to impersonate any database user, with full administrative rights</td>
<td>GRANT SET USER (ANY)</td>
</tr>
<tr>
<td></td>
<td>TO user_ID [,...]</td>
</tr>
<tr>
<td></td>
<td>WITH ADMIN OPTION</td>
</tr>
<tr>
<td>System privilege to impersonate any database user, with administrative rights only</td>
<td>GRANT SET USER (ANY)</td>
</tr>
<tr>
<td></td>
<td>TO user_ID [,...]</td>
</tr>
<tr>
<td></td>
<td>WITH ADMIN ONLY OPTION</td>
</tr>
<tr>
<td>System privilege to impersonate any database user, with no administrative rights</td>
<td>GRANT SET USER (ANY)</td>
</tr>
<tr>
<td></td>
<td>TO user_ID [,...]</td>
</tr>
<tr>
<td></td>
<td>WITH NO ADMIN OPTION</td>
</tr>
<tr>
<td>System privilege to impersonate specified users</td>
<td>GRANT SET USER (target_users_list)</td>
</tr>
<tr>
<td></td>
<td>TO user_ID [,...]</td>
</tr>
<tr>
<td>System privilege to impersonate any member of specified roles</td>
<td>GRANT SET USER (ANY WITH ROLES target_roles_list)</td>
</tr>
<tr>
<td></td>
<td>TO user_ID [,...]</td>
</tr>
</tbody>
</table>
Grant Type | Statement
---|---
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 these 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
```

See also
- *GRANT SET USER Statement* on page 272

Start Impersonating Another User

Allow a user to assume the exact roles and system privileges (impersonate) of another user. Once begun, impersonation remains in effect until it is terminated or the current session ends.

Prerequisites

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

Note: For the purposes of meeting 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. For example, User1 is granted Role1 with the WITH ADMIN OPTION clause, User2 is granted Role1 with the WITH ADMIN ONLY clause, and User3 is granted Role1 with the WITH NO ADMIN OPTION clause. User1 and User2 are said to be granted Role1 with similar administrative rights. User1 and User2 are also said to be granted Role1 with higher administrative rights that User3.

• If the target user has been granted a system privilege which supports extensions, the clauses used to grant the system privilege to the impersonator are a super-set of those used for the target user. Currently, only the SET USER and CHANGE PASSWORD system privileges support extensions.

Note:
• 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, Role1, Role2, respectively, while the target list grants of the target user contain User1, 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 themselves. Therefore, if the target user has been 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.

Task
Validation of the at-least criteria occurs when the SETUSER command is executed. If a user fails to meet any of the at-least requirements, a permission denied message appears, and the impersonation does not begin.
At a command prompt, type:

```plaintext
SETUSER userID
```

**See also**

- *SETUSER Statement on page 292*

**Verify 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 this command on a machine on which the `SETUSER` command was issued:

```plaintext
SELECT CURRENT USER
```

The name of the user the machine recognizes as the currently logged in user appears. If it is the expected user for the machine, no impersonation is active on the machine. If an unexpected user name appears, it represents the user currently being impersonated on the machine.

**Example**

On a connection where Joe is logged in, execute:

```plaintext
> select current user
> go
current user
Joe
(1 row affected)

> setuser mary
> go
> select current user
> go
current user
Mary
```

**Stop Impersonating Another User**

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

**Prerequisites**

The `SETUSER` command can be used to terminate an impersonation only from the connection where the `SETUSER` command was originally issued to start the impersonation.
Task
At a command prompt, type:

SETUSER

See also
•  SETUSER Statement on page 292

Revoking the SET USER System Privilege from a User
Remove the ability of a user to impersonate other users and administer the system privilege.

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

Task
The SET USER system privilege can be granted to a user multiple times, using different clauses. For example, UserA 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 form of the clause used for the GRANT must be used to revoke it. Continuing with the example, if the system privilege is revoked from UserA using the ANY clause, the grant with the target_users_list clause remains in effect. The net effect is that UserA is now limited to impersonating users on the target_users_list. Alternately, if the system privilege is revoked from UserA using the target_users_list clause, the grant with the ANY clause remains in effect. The net effect in this scenario is that UserA can continue to impersonate any user in the database.

Note: These examples assume UserA meets all criteria for successful impersonation.

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

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

**Example:**

These statements removes the ability for *Sam* to impersonate any database user:

```sql
REVOKE SET USER (ANY) FROM Sam
or
REVOKE SET USER FROM Sam
```

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

```sql
REVOKE ADMIN OPTION FOR SET USER (ANY) FROM Frank
```

This statement removes the ability of *Bob* and *Jeff* to impersonate *Mary*, *Joe*, or *Sue* only.

```sql
REVOKE SET USER (Mary, Joe, Sue) FROM Bob, Jeff
```

This statement removes the ability of *Mary* to impersonate any member of the *Sales1* role:

```sql
REVOKE SET USER (ANY WITH ROLES Sales1) FROM Mary
```

This statement removes the ability of *Sarah* to impersonate *Joe* or *Sue*, or any member of the *Sales2* role:

```sql
REVOKE SET USER (Joe, Sue), (ANY WITH ROLES Sales2) FROM Sarah
```

This statement removes the ability of *Joan* to impersonate any member of the *Marketing1* or *Marketing2* roles:

```sql
REVOKE SET USER (ANY WITH ROLES Marketing1, Marketing2) FROM Joan
```

**See also**

- *REVOKE SET USER Statement* on page 284

**Users**

User management includes the creation and deletion of user IDs, as well as management of passwords.
**DBA User**

The DBA user is the default user created when a new SAP Sybase IQ database is created. The password for the DBA user is initially set to "sql." To override the default user name or password during creation, use the `CREATE DATABASE` statement with the DBA USER or DBA PASSWORD clause.

If you elect not to override the default password during creation, it is strongly recommended that you do so as soon as possible thereafter.

By default, the DBA user is automatically granted administrative rights on the SYS_AUTH_DBA_ROLE role, which in turn is granted the SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE roles. It is the union of these roles which grants the DBA user all system and object-level privileges in the database, and allows DBA to carry out any activity in the database: create tables, change table structures, create new user IDs, revoke privileges from users, and so on.

To ensure database security and accountability, avoid using generic names like "dba" as the first user ID. Use a real user's login name with a strong password instead.

*Users Granted 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. However, the SAP Sybase IQ documentation assumes that the DBA user is the database administrator, and all underlying roles and system privileges remain as granted by default.

To guard against loss of password by the active DBA user, create one or more extra DBA accounts (with a randomly 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.

*Adding New Users*

The DBA can add new users to the database. New users are then granted privileges to carry out authorized tasks on the database. Although DBA responsibilities may be handed over to other user IDs, the DBA is responsible for the overall management of the database by virtue of the SYS_AUTH_DBA_ROLE role.

The DBA can then create database objects and assign ownership of these objects to other user IDs.

*DBA User ID in Case-Sensitive Databases*

User IDs and passwords are actually database objects.
**Changing the DBA Password**
Change the password for the DBA user. The default password for DBA user for all databases is `sql`. You should change this password to prevent unauthorized access to your database.

**Prerequisites**
Requires the CHANGE PASSWORD system privilege.

**Note:** If you are using dbisql, it is suggested that you place your permission grants into a command file for reference so you can modify and re-run it if necessary, to re-create the permissions.

**Task**
To change a user password, execute:

```
ALTER USER userID IDENTIFIED BY password
```

Example:

This example changes the DBA user password to `S&ecureAdmln`:

```
ALTER USER DBA IDENTIFIED BY S&ecureAdmln
```

**See also**
- *Case-sensitivity of User IDs and Passwords* on page 122
- *ALTER USER Statement* on page 242

**Super-User**
Super-users are users that can exercise any system privilege and administer any role; they can perform any privileged operation in the system. Role-based security does not require a super-user to maintain the database, and the DBA user may not be a super-user.

By default, the DBA user can exercise any system privilege, but since it may not be able to administer all user-defined roles, it is not considered a true super-user. SAP Sybase IQ does not automatically create a super-user for a new or migrated database.

To create a super-user, create a user and grant it the `SYS_AUTH_DBA_ROLE` compatibility role.

**Note:** If you migrated `SYS_AUTH_DBA_ROLE`, you must manually grant all of the underlying default system privileges of `SYS_AUTH_DBA_ROLE`, with administration rights, to create the super-user.

Once the super-user is created, going forward, to maintain the super-user status, all new user-extended and user-defined roles must be granted to the super-user, with administrative rights.
To allow the DBA user to act as a super-user, all new user-extended and user-defined roles must be granted to the DBA user, with administrative rights.

Administrative rights can be granted in the form of a role administrator or a global role administrator.

**Increase Password Security**

Passwords are an important part of any database security system. There are several options available to increase password security.

*Implement a Login Policy*

Use a login policy to 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*.

*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*.

*Implement Password Rules*

You can 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*.

**Case-sensitivity of User IDs and Passwords**

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

In SAP Sybase IQ and SQL Anywhere, all passwords in newly-created databases are case-sensitive, regardless of the case-sensitivity of the database. The default user ID is DBA and the password for this user is lowercase sql.

When you rebuild an existing database, SAP Sybase IQ and SQL Anywhere determine the case-sensitivity of the password as follows:

- If the database 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.

In SAP® Sybase Adaptive Server Enterprise, the case-sensitivity of user IDs and passwords follows the case-sensitivity of the server.

Creating a New User
Create a new user ID.

Prerequisites
Requires the MANAGE ANY USER system privilege.

Task
To create a new user, execute:

```
CREATE USER userID
IDENTIFIED BY password
```

Example:

This statement adds user ID Joe to a database with password welcome:

```
CREATE USER Joe
IDENTIFIED BY welcome
```

See also
- CREATE USER Statement on page 255

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.

Task
If the user being delete has any external logins defined, the external logins are deleted as part of the process. However, any related objects on remote servers are not removed.

To delete a user, execute:

```
DROP USER userID
```

Note: When dropping a user, any permissions granted by this user will be removed.
Note: If the user being deleted owns any objects in the database, when the **DROP USER** command is issue, the following error message appears, and the command fails:

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

Example:

This statement drops user ID **Joe** from the database:

```sql
DROP USER Joe
```

**See also**

- *DROP USER Statement* on page 260

---

## Changing a User's Password

Change the password of another user.

### Prerequisites

Requires the CHANGE PASSWORD system privilege.

### Task

You can set up password rules (**MIN_PASSWORD_LENGTH** option) and verify that any new password assigned complies with them (**VERIFY_PASSWORD_FUNCTION** option). For example, you might require that passwords must include one digit or cannot be the user ID.

To change a user password, execute:

```sql
ALTER USER  user_ID
       IDENTIFIED BY  password
```

Example:

This statement assigns the new password **P&ssW0rd** to user **M_Smith**:

```sql
ALTER USER M_Smith IDENTIFIED BY P&ssW0rd
```

**See also**

- *Case-sensitivity of User IDs and Passwords* on page 122
- *ALTER USER Statement* on page 242
- *VERIFY_PASSWORD_FUNCTION Option* on page 299
- *MIN_PASSWORD_LENGTH Option* on page 302
Converting a User-Extended Role Back to a User

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

Prerequisites
Requires administrative rights over the user-extended role being converted.

Task
The user retains any login privileges, system privileges and roles granted to the user-extended role. The user remains as the owner of the objects that were created after the user was extended to act as a role. Any members of the user-extended role are immediately revoked.

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 converting a user-extended role back to a user, all dependent roles of the user-extended role must continue to meet this minimum requirement, or the conversion fails.

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

<table>
<thead>
<tr>
<th>Convert Condition</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role has not been granted any members.</td>
<td>DROP ROLE FROM USER <code>role_name</code></td>
</tr>
<tr>
<td>Role has been granted members.</td>
<td>DROP ROLE FROM USER <code>role_name</code> WITH REVOKE</td>
</tr>
</tbody>
</table>

See also
•  *DROP ROLE Statement* on page 258

Permanently 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 Sybase IQ server.

Prerequisites
•  Requires the MANAGE ANY LOGIN POLICY system privilege to create/alter the login policy.
•  Requires the MANAGE ANY USER system privilege to assign the login policy to users.
Security Management

Task

1. Create a login policy with the LOCKED option set to ON.
2. Execute the `ALTER USER` command to assign the login policy to a user account to be disabled.

   **Note:** You cannot specify multiple user names in the same `ALTER USER` command when assigning a login policy to users.

Examples:

This command creates a new login policy named `lp_locked_users` with the LOCKED option set to ON:

```
CREATE LOGIN POLICY lp_locked_users locked=ON
```

These commands assign the `lp_locked_users` login policy to users John and Mary:

```
ALTER USER john LOGIN POLICY lp_locked_users
ALTER USER mary LOGIN POLICY lp_locked_users
```

John and Mary can no longer log in.

See also

- Automatic Unlocking of User Accounts on page 127
- `ALTER USER Statement` on page 242
- `CREATE LOGIN POLICY Statement` on page 248

Unlocking User Accounts

Unlock a user account.

Prerequisites

Requires the MANAGE ANY USER system privilege.

Task

The manner in which you unlock an account depends on how it was originally locked.

1. If a user account is locked because it is assigned to a login policy with the locked option set to ON, reassign the user to a login policy with the locked option set to OFF.

2. If a user account is locked because it has exceeded the `MAX_FAILED_LOGIN_ATTEMPTS` or `MAX_DAYS_SINCE_LOGIN`, issue the `ALTER USER` statement with the RESET LOGIN POLICY option. Forcing the reset of the login policy reverts the settings of the user's login to the original values in the login policy. This usually clears all locks that are implicitly set due to the user exceeding the failed logins or exceeding the maximum number of days since the last login.
Note: Resetting the values in the login policy assigned to a user does not reset the values for all users assigned the same login policy.

Example

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

```
ALTER USER john LOGIN POLICY lp
```

Assuming the account of John is locked because he either exceeded the MAX_FAILED_LOGIN_ATTEMPTS or MAX_DAYS_SINCE_LOGIN, this example forces the reset of the values in the login policy currently assigned to John:

```
ALTER USER john RESET LOGIN POLICY
```

See also

- *Automatic Unlocking of User Accounts* on page 127
- *ALTER LOGIN POLICY Statement* on page 233
- *ALTER USER Statement* on page 242

Automatic Unlocking of User Accounts

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

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 must be manually unlocked by a user granted the MANAGE ANY USER system privilege. However, if all users with the MANAGE ANY USER system privilege are locked out due to failed login attempts, a potential lock-down of some or all the database services can occur.

To prevent this scenario, use these login policy options:

- **ROOT_AUTO_LOCK_TIME** – defines automatic unlocking period for users with the MANAGE ANY USER system privilege. You can set root_auto_lock_time to a small value (for example, 15 minutes) in the root login policy. There is a server-imposed upper limit of a few hours.

- **AUTO_UNLOCK_TIME** – defines the automatic unlocking period for all other users. Set AUTO_UNLOCK_TIME to UNLIMITED (default) in any login policy, including the root login policy.

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

Based on the permissions granted to a user, one of these login policy options is verified at the time of unlocking. 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 log in and if the user has equaled or exceeded the specified automatic unlock period, the user is allowed to login and the FAILED_LOGIN_ATTEMPTS counter is reset to zero.

See also
• Minimum Number of Role Administrators on page 20
• Unlocking User Accounts on page 126
• Permanently Locking a User Account on page 125
• ALTER LOGIN POLICY Statement on page 233
• ALTER USER Statement on page 242

Login Policies

A login policy defines the rules that SAP Sybase 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.

Warning! Migrating databases from version 12.7 removes existing login management settings. You must re-create them after migration.

Modifying the Root Login Policy

You can modify the option values for the root login policy, but you cannot drop the policy.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
Each new database is created with a default login policy, called the root policy. When you create a user account without specifying a login policy, the user becomes part of the root login policy.

To modify the options of the root login policy, execute:

```
ALTER LOGIN POLICY ROOT {login_policy_options}
```

See also
• ALTER LOGIN POLICY Statement on page 233
• Login Policy Options on page 249
• Multiplex Login Policy Configuration on page 239
• LDAP Login Policy Options on page 237
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.

Task
Login policy names must be unique. An error message appears if the login policy name already exists.
To create a new login policy, execute:

\[
\text{CREATE LOGIN POLICY policy_name \{login_policy_options\}}
\]

Example:

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

\[
\text{CREATE LOGIN POLICY Test1 password_life_time=60}
\]

See also
• CREATE LOGIN POLICY Statement on page 248
• Login Policy Options on page 249
• Multiplex Login Policy Configuration on page 239
• LDAP Login Policy Options on page 237

Modifying an Existing Login Policy

Use Interactive SQL to change the options for an existing login policy.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
To alter the options of an existing login policy, execute:

\[
\text{ALTER LOGIN POLICY policy-name \{login_policy_options\}}
\]

Example:

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

\[
\text{ALTER LOGIN POLICY Test1 locked=on max_connections=5}
\]
Deleting a Login Policy
You cannot delete the root login policy or one currently assigned to a user.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
1. Verify that no users are currently assigned the login policy to be dropped.
2. To drop a login policy, execute:
   ```
   DROP LOGIN POLICY policy_name
   ```

Assigning a Login Policy When Creating a New User
If you do not assign a login policy when creating a user account, the account is assigned the root login policy.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
Assign a login policy other than the root login policy when creating a new user. A user can be assigned only one login policy at a time.
To assign a login policy, execute:
```sql
CREATE USER userID
[ [IDENTIFIED BY password ]
[ LOGIN POLICY policy-name ]
```

Note: You cannot specify multiple user IDs in the same CREATE USER command when assigning a login policy to users.

Example:
This statement creates a user called Joe with the password welcome, and assigns the login policy Test2:
CREATE USER Joe
IDENTIFIED BY welcome
LOGIN POLICY Test2

See also
• CREATE USER Statement on page 255

Assigning a Login Policy to an Existing User

Use Interactive SQL to assign a login policy to an existing user.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
1. To change the login policy assigned to a user, execute:
   
   ```
   ALTER USER userID
   LOGIN POLICY policy_name
   ```

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

See also
• ALTER USER Statement on page 242

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 set value when attempting to log in, the 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.
Preventing Connection After Failed Login Attempts

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

Prerequisites

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

Task

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 assigned the login policy.

1. To set the MAX_FAILED_LOGIN_ATTEMPTS option, either create a new login policy, or modify an existing one.
2. Define a value for the MAX_FAILED_LOGIN_ATTEMPTS option.
3. 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 five 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 five failed attempts:

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

This example assigns the login policy lp to user John:

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

Once John is assigned the lp login policy, he cannot log in if he enters invalid credentials five times in sequence.

See also

- ALTER LOGIN POLICY Statement on page 233
- ALTER USER Statement on page 242
- CREATE LOGIN POLICY Statement on page 248
- Login Policy Options on page 235
Creating a DBA Recovery Account

Create a DBA recovery account for production systems. The DBA recovery account is a backup, in case you lose the original DBA account password.

1. Create one or more extra DBA accounts, using randomly generated user names and passwords.
2. Lock the credentials in a secure location.

See also

• CREATE USER Statement on page 255

Logging in with a DBA Recovery Account

Log in using the DBA recovery account, and reset the original DBA account password.

1. Retrieve the DBA recovery account user name and password from the secure location.
2. Log in using the recovery account.
3. Reset the original DBA account password.
4. Return the DBA recovery account credentials to their secure location.

Manage Connections Using Stored Procedures

There are several stored procedures for managing user connections.

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

<table>
<thead>
<tr>
<th>Stored Procedure</th>
<th>Purpose</th>
<th>System Privilege Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>sa_get_user_status</td>
<td>Retrieve the current status of all existing users</td>
<td>Requires the 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.</td>
</tr>
<tr>
<td>sp_expireallpasswords</td>
<td>Cause all user passwords to expire immediately</td>
<td>MANAGE ANY USER system privilege</td>
</tr>
</tbody>
</table>
### Stored Procedure

<table>
<thead>
<tr>
<th>Stored Procedure</th>
<th>Purpose</th>
<th>System Privilege Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp_iqaddlogin</td>
<td>Add users, define their passwords, specify login policy, and password expiry on next login</td>
<td>MANAGE ANY USER system privilege</td>
</tr>
<tr>
<td>sp_iqcopypolloginpolicy</td>
<td>Create a new login policy by copying an existing one</td>
<td>MANAGE ANY LOGIN POLICY system privilege</td>
</tr>
<tr>
<td>sp_iqdroplogin</td>
<td>Drop the specified user</td>
<td>MANAGE ANY USER system privilege</td>
</tr>
<tr>
<td>sp_iqmodifylogin</td>
<td>Assign a given user to a login policy</td>
<td>MANAGE ANY USER system privilege</td>
</tr>
<tr>
<td>sp_iqmodifyadmin</td>
<td>Set an option on a named login policy to a certain value</td>
<td>MANAGE ANY LOGIN POLICY system privilege</td>
</tr>
<tr>
<td>sp_iqpassword</td>
<td>Change your own or another user’s password</td>
<td>All users can run sp_iqpassword to change their own password. However, the CHANGE PASSWORD system privilege is required to change the password of another user.</td>
</tr>
</tbody>
</table>

### See also
- *sp_expireallpasswords system procedure* on page 320
- *sp_iqcopypolloginpolicy Procedure* on page 331
- *sp_iqdroplogin Procedure* on page 342
- *sp_iqmodifyadmin Procedure* on page 348
- *sp_iqmodifylogin Procedure* on page 349
- *sp_iqpassword Procedure* on page 361
- *sp_iqaddlogin Procedure* on page 323
- *sa_get_user_status system procedure* on page 314

### 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.

You can set database options using the SET OPTION statement.

Resources You Can Manage

- **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_CONNECTION** – sets the number of processing threads available to a connection for use in IQ operations.
- **TEMP_CACHE_MEMORY_MB** – sets the size of the cache for the SAP Sybase IQ temporary store. (The server option `-iqtcache` is the recommended way to set the temp cache size.)
- **QUERY_TEMP_SPACE_LIMIT** – limits the amount of temporary dbspace available to any one query.
- **QUERY_ROWS_RETURNED_LIMIT** – tells the query optimizer to reject queries that might consume too many resources. If the optimizer estimates that the result set from the query will exceed the value of this option, the optimizer rejects the query and returns an error message.

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

- **JAVA_HEAP_SIZE** – sets the maximum size (in bytes) of the memory allocated to Java applications on a per connection basis.
- **MAX_CURSOR_COUNT** – limits the number of cursors for a connection.
- **MAX_STATEMENT_COUNT** – limits the number of prepared statements for a connection.

Database option settings are not inherited through the role structure.

See also

- *SET OPTION Statement* on page 290
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, defining privileges directly on tables has limitations. 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 wish to give sales representatives privileges on a table containing descriptions of sales calls, but limit update privileges to their own calls.

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 limit a user to see only the rows of a table that he or she have 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:

   CONNECT "DBA"
   IDENTIFIED by sql;
   GRANT CONNECT
   TO SalesManager
   IDENTIFIED BY sales

   Enclose DBA in quotation marks because it is a SQL keyword.

2. Define a view that looks only at sales employees:

   CREATE VIEW emp_sales AS
   SELECT EmployeeID, GivenName, Surname
   FROM "DBA".Employees
   WHERE DepartmentID = 200

   Identify the table as "DBA".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.

3. Give SalesManager privilege to look at the view:

   GRANT SELECT
   ON emp_sales
   TO SalesManager

   Use the same command to grant privilege on a view as to grant privilege on a table.

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 sales;
   SELECT * FROM "GROUPO".emp_sales;
   SELECT * FROM "GROUPO".order_summary;

   No privileges have been granted to SalesManager to look at the underlying tables. These commands produce privilege errors:

   SELECT * FROM "DBA".Employees;
   SELECT * FROM "DBA".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.

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, a derived table, or a 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
```

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.

**Warning!** Do not delete views owned by the dbo user ID, which owns system objects. Deleting such views or changing them into tables may cause unexpected problems.

### Procedures Provide Tailored Security

Procedures restrict the actions 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/ALTER PROCEDURE` statement.
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.

To allow users with specific privileges to administer certain tasks using SAP Sybase IQ system procedures:

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 a new user is created who is to be granted authorized tasks, grant the role created for each authorized task to the user.

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
Requires the MANAGE ANY USER or EXECUTE ANY PROCEDURE system privilege.

Task
To grant user user1 the MANAGE ANY USER system privilege and privileges to execute procedures related to user administration:

1. Create a role USER_ADMIN_GRP:

   ```sql
   CREATE ROLE USER_ADMIN_GRP
   ```

2. Grant the MANAGE ANY USER system privilege to the USER_ADMIN_GRP role:

   ```sql
   GRANT MANAGE ANY USER TO USER_ADMIN_GRP
   ```

3. Grant EXECUTE privilege on SAP Sybase IQ stored procedures for user administration to USER_ADMIN_GRP:

   ```sql
   GRANT EXECUTE on sp_iqaddlogin to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqcopypolicy to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqdroplogin to USER_ADMIN_GRP
   GRANT EXECUTE on sp_iqmodifyadmin to USER_ADMIN_GRP
   ```
4. Grant the USER_ADMIN_GRP role to user1. user1 inherits the MANAGE ANY USER system privilege and the ability to execute the assigned IQ procedures through membership in USER_ADMIN_GRP role.

GRANT ROLE USER_ADMIN_GRP TO user1

Related Stored Procedures for Role Access
You may create roles that grant privileges for various related stored procedures.

<table>
<thead>
<tr>
<th>Role Name</th>
<th>System Privilege Granted</th>
<th>Stored Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATOR_GRP</td>
<td>BACKUP DATABASE</td>
<td>sp_iqbackupdetails</td>
</tr>
<tr>
<td></td>
<td>DROP CONNECTION</td>
<td>sp_iqbackupsummary</td>
</tr>
<tr>
<td></td>
<td>CHECKPOINT</td>
<td>sp_iqconnection</td>
</tr>
<tr>
<td></td>
<td>MONITOR</td>
<td>sp_iqsysmon</td>
</tr>
<tr>
<td></td>
<td>ACCESS SERVER LS</td>
<td></td>
</tr>
<tr>
<td>SPACEADMIN_GRP</td>
<td>MANAGE ANY DBSPACE</td>
<td>sp_iqdbspace</td>
</tr>
<tr>
<td></td>
<td>ACCESS SERVER LS</td>
<td>sp_iqdbspaceinfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqdbspaceobjectinfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqemptyfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqestdbspaces</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqobjectinfo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sp_iqspaceused</td>
</tr>
</tbody>
</table>

See also
- *sp_iqbackupdetails Procedure* on page 324
- *sp_iqbackupsummary Procedure* on page 326
- *sp_iqconnection Procedure* on page 327
- *sp_iqdbspace Procedure* on page 332
- *sp_iqdbspaceinfo Procedure* on page 334
- *sp_iqdbspaceobjectinfo Procedure* on page 338
- *sp_iqemptyfile Procedure* on page 343
- *sp_iqestdbspaces Procedure* on page 344
- *sp_iqfile Procedure* on page 345
- *sp_iqobjectinfo Procedure* on page 349
- *sp_iqspaceused Procedure* on page 352
Confidentiality of Data

You can secure communications between a client and the SAP Sybase IQ server, or between an SAP Sybase IQ client and the database server using Transport Layer Security (TLS).

SAP Sybase IQ allows you to encrypt your database or columns.

Support of FIPS encryption, Kerberos authentication, and column encryption is included in the separately licensed SAP Sybase IQ Advanced Security Option.

See also

- Column Encryption in SAP Sybase IQ on page 192
- FIPS Support in SAP Sybase IQ on page 191

Database encryption and decryption

You can use database encryption to make it more difficult for someone to decipher the data in your database. You can choose to secure your database either with simple or with strong encryption.

Note: If your database is encrypted, compressing it with a tool such as WinZip does not result in a file that is significantly smaller than the original database file.

Simple encryption and strong encryption

Simple encryption

Simple encryption is equivalent to obfuscation and makes it more difficult for someone using a disk utility to look at the file to decipher the data in your database. Simple encryption does not require a key to encrypt the database.

Strong encryption

Strong database encryption technology makes a database inoperable and inaccessible without a key (password). An algorithm encodes the information contained in your database and transaction log files so they cannot be deciphered.

In SAP Sybase IQ, the database administrator has control over four aspects of strong encryption, including:

- strong encryption status
- encryption key
- protection of the encryption key
- encryption algorithm
Supported strong encryption algorithms
The algorithm used to implement SAP Sybase IQ strong encryption is AES: a block encryption algorithm chosen as the new Advanced Encryption Standard (AES) for block ciphers by the National Institute of Standards and Technology (NIST).

You can also specify a separate FIPS-approved AES module for strong encryption using the AES_FIPS (128-bit) or AES256_FIPS (256-bit) type. When the database server is started with the -fips option, you can run databases encrypted with AES, AES256, AES_FIPS, or AES256_FIPS strong encryption, but not databases encrypted with simple encryption. Unencrypted databases can also be started on the server when -fips is specified.

The SAP Sybase IQ security option must be installed on any computer used to run a database encrypted with AES_FIPS or AES256_FIPS.

FIPS-certified encryption is not available on all platforms. For a list of supported platforms, see http://www.sybase.com/detail?id=1061806.

Note: Separately licensed component required.

FIPS-certified encryption requires a separate license. All strong encryption technologies are subject to export regulations.

Database encryption methods

• **To create an encrypted database** – You can use the following:
  • The Initialization utility (iqinit) in combination with various options to enable strong encryption.
    The iqinit utility -ep and -ek options create a database with strong encryption, allowing you to specify the encryption key in a prompt box or on the command line. The iqinit -ea option sets the encryption algorithm to AES or AES256 (or to AES_FIPS or AES256_FIPS for the FIPS-certified module).
  • CREATE DATABASE statement.

• **To encrypt an existing database** – Although you cannot simply turn strong encryption on or off in an existing database, you can use one of the following to implement strong encryption:
  • Rebuild (unload/reload) an existing database and change the encryption status at that time. You can rebuild the database to unload all the data and schema of an existing database. This creates a new database (at which point you can change a variety of settings including strong encryption status), and reloads the data into the new database.
    You need to know the key to unload a strongly encrypted database. To rebuild (unload/reload) a database, use one of the following methods:
    • The Unload utility (dbunload)
    The Unload utility (dbunload) with options to create a new database with strong encryption. The -an option creates a new database. To specify strong encryption and the encryption key in a prompt box or on the command line use the -ep or -ek
option. The -ea option sets the encryption algorithm to AES or AES256 (or to AES_FIPS or AES256_FIPS for the FIPS-certified module).

- The UNLOAD and RELOAD statements
- The Unload Database Wizard.
- You can use the CREATE ENCRYPTED DATABASE statement or the CREATE ENCRYPTED FILE statement.
- **To encrypt tables, columns, and materialized views** – See Column and table encryption.

See also
- *Column and table encryption* on page 147

**Comparison of CREATE ENCRYPTED DATABASE and CREATE ENCRYPTED FILE statements**

You should use the CREATE ENCRYPTED DATABASE statement when you have an existing database that you want to encrypt. Use CREATE ENCRYPTED FILE statement only in the case where you have a database you want to encrypt that requires recovery.

You cannot be connected to the database you are encrypting when you execute the statement.

The CREATE ENCRYPTED FILE and CREATE ENCRYPTED DATABASE statements differ from each other as follows:

- The CREATE ENCRYPTED FILE statement must be executed against each of the database-related files independently (transaction log, transaction log mirror, dbspaces, if any), whereas the CREATE ENCRYPTED DATABASE statement automatically encrypts all the database-related files.
- The CREATE ENCRYPTED DATABASE statement cannot be used on a database requiring recovery; the CREATE ENCRYPTED FILE statement can.
- The CREATE ENCRYPTED DATABASE statement cannot be used inside procedures, triggers, or batches. The CREATE ENCRYPTED FILE statement can.
- The CREATE ENCRYPTED DATABASE statement supports the SIMPLE encryption algorithm, but the CREATE ENCRYPTED FILE statement does not.

**Creating an encrypted database (SQL)**

You can encrypt a database during creation by using the ENCRYPTED clause with the CREATE DATABASE statement.

**Prerequisites**

By default, you must have the SERVER OPERATOR system privilege. The required privileges can be changed by using the -gu database server option.

**Task**

This task is different from encrypting an existing database. To encrypt an existing database, use the CREATE ENCRYPTED DATABASE statement.
Warning! 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.

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.

An encrypted database is created.

Creating an encrypted database (iqinit utility)
You can create an encrypted database using the iqinit utility.

Prerequisites
There are no prerequisites for this task.

Task

Warning! 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.

Run the iqinit utility to create a database.

- To encrypt the database with simple encryption, include the -ea simple option.
- To encrypt the database with strong encryption, include -ek or -ep options to specify the encryption key.

An encrypted database is created.

Next
When starting or connecting to the database, you must specify the encryption key.

Creating an encrypted copy of an existing database (SQL)
You can create an encrypted copy of a database by using the CREATE ENCRYPTED DATABASE statement. This statement creates a copy of the file (in this case, in encrypted form), and does not overwrite the original database file.

Prerequisites
By default, you must have the SERVER OPERATOR system privilege to execute the CREATE ENCRYPTED DATABASE statement. The required privileges can be changed by using the -gu database server option.

The database you are encrypting must not be running.
**Task**

**Warning!** 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.

1. In Interactive SQL, connect to an existing database, other than the one you are encrypting.
2. Encrypt the database using the `CREATE ENCRYPTED DATABASE` statement.

When you execute a `CREATE ENCRYPTED DATABASE` statement, you do not encrypt (overwrite) the file; you create a copy of the file in encrypted form. If there are transaction logs, transaction log mirrors, or dbspaces associated with the database, encrypted copies of those files are made as well.

**Decrypting a database (SQL)**

You can decrypt a database using the `CREATE DECRYPTED DATABASE` statement. This statement creates a copy of the file (in decrypted form) and does not overwrite the original database file.

**Prerequisites**

By default, you must have the `SERVER OPERATOR` system privilege to execute the `CREATE DECRYPTED TABLE DATABASE` statement. The required privileges can be changed by using the `-gu database` server option.

The database you are encrypting must not be running.

**Task**

If you have a database that requires recovery and you want to decrypt it to send it to Technical Support, you must use the `CREATE DECRYPTED FILE` statement. Any database-related files such as transaction logs and transaction log mirrors, and dbspace files, must also be decrypted using this statement.

1. In Interactive SQL, connect to a database other than the one you want to decrypt.
2. Execute a `CREATE DECRYPTED DATABASE` statement.

When you execute a `CREATE DECRYPTED DATABASE` statement, you do not decrypt (overwrite) the file; you create a copy of the file in decrypted form. If there are transaction logs, transaction log mirrors, or dbspaces associated with the database, decrypted copies of those files are made as well.

**Encryption keys**

It is best to choose an encryption key value that cannot be easily guessed. The key can be of arbitrary length, but generally the longer the key, the better because a shorter key is easier to
guess than a longer one. 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.

You must supply this key each time you want to start the database. Lost or forgotten keys result in completely inaccessible databases.

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. Clients are required to specify the key each time they start the database. If the database administrator starts the database, clients never need to have access to the key.

**Warning!** 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.

**Changing the encryption key for a database**
You can change the encryption key for an encrypted database, or for a database for which table encryption has been enabled, by using the CREATE ENCRYPTED DATABASE statement. Changing the encryption key does not overwrite the existing file, but creates a copy of the file encrypted with the new key.

**Prerequisites**
By default, you must have the SERVER OPERATOR system privilege to execute the CREATE ENCRYPTED DATABASE statement. The required privileges can be changed by using the -gu database server option.

**Task**
Change the encryption key for an encrypted database using the CREATE ENCRYPTED DATABASE statement.

The encryption key is changed.

**Security and performance issues**
Performance of SAP Sybase IQ 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.
Column and table encryption
If you only want to encrypt portions of your database, you can choose to encrypt columns or tables.

Column encryption can be performed on any column in any table at any time. Table encryption requires that the database have table encryption enabled. Table encryption is enabled at database creation (initialization) time.

- **To encrypt tables** – You can use the following:
  - Initialization utility (iqinit).
  - CREATE DATABASE statement.
  - ALTER DATABASE statement.
  - CREATE ENCRYPTED TABLE DATABASE statement.
- **To encrypt columns** – ENCRYPT function.
- **To encrypt materialized views** – ALTER MATERIALIZED VIEW statement.

Column encryption
To encrypt columns in your database, use the ENCRYPT function. The ENCRYPT function uses the same AES strong encryption algorithm that is used for database encryption to encrypt values that are passed to it.

Encrypted data can be decrypted with the DECRYPT function. You must use the same key that was specified in the ENCRYPT function. Both of these functions return LONG BINARY values. If you require a different data type, you can use the CAST function to convert the value to the required data type.

The ENCRYPT and DECRYPT functions also support raw encryption. You can encrypt data inside the database server into a format that can be exported and decrypted outside of the server.

If database users need to access the data in decrypted form, but you do not want them to have access to the encryption key, you can create a view that uses the DECRYPT function. This allows users to access the decrypted data without knowing the encryption key. If you create a view or stored procedure that uses the table, you can use the SET HIDDEN parameter of the ALTER VIEW and ALTER PROCEDURE statements to ensure that users cannot access the encryption key by looking at the view or procedure definition.

Column encryption example
The following example uses triggers to encrypt a column that stores passwords in a table called user_info. The user_info table is defined as follows:

```sql
CREATE TABLE user_info (  
    employee_ID INTEGER NOT NULL PRIMARY KEY,  
    user_name CHAR(80),  
    user_pwd CHAR(80) );
```
Two triggers are added to the database to encrypt the value in the user_pwd column, either when a new user is added or an existing user's password is updated.

- The encrypt_new_user_pwd trigger fires each time a new row is added to the user_info table:

```sql
CREATE TRIGGER encrypt_new_user_pwd
BEFORE INSERT
ON user_info
REFERENCING NEW AS new_pwd
FOR EACH ROW
BEGIN
    SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
END;
```

- The encrypt_updated_pwd trigger fires each time the user_pwd column is updated in the user_info table:

```sql
CREATE TRIGGER encrypt_updated_pwd
BEFORE UPDATE OF user_pwd
ON user_info
REFERENCING NEW AS new_pwd
FOR EACH ROW
BEGIN
    SET new_pwd.user_pwd=ENCRYPT(new_pwd.user_pwd, '8U3dkA');
END;
```

Add a new user to the database:

```sql
INSERT INTO user_info
VALUES ( '1', 'd_williamson', 'abc123');
```

If you issue a SELECT statement to view the information in the user_info table, the value in the user_pwd column is binary data (the encrypted form of the password) and not the value abc123 that was specified in the INSERT statement.

If this user's password is changed, then the encrypt_updated_pwd trigger fires and the encrypted form of the new password appears in the user_pwd column.

```sql
UPDATE user_info
SET user_pwd='xyz'
WHERE employee_ID='1';
```

The original password can be retrieved by issuing the following SQL statement. This statement uses the DECRYPT function and the encryption key to decrypt the data, and the CAST function to convert the value from a LONG BINARY to a CHAR value:

```sql
SELECT CAST ( 
    DECRYPT( user_pwd, '8U3dkA' ) 
AS CHAR(100))
FROM user_info
WHERE employee_ID = '1';
```

**Raw encryption**

Raw encryption allows you to encrypt data inside the database server into a format that can be exported and decrypted outside of the database server. The encrypted format is referred to as raw. To encrypt data in the raw format, you must specify the encryption key, the initialization
vector, and optionally a padding format. To decrypt the data, you must specify the same parameter values.

You can also use the DECRYPT function to decrypt the data inside the database server.

Raw encryption is useful when:

- **You want to prevent database users from having access to the data** – You can use raw encryption to encrypt sensitive data that you do not want even your database administrators to have access to, and then decrypt the data using a client application without the use of the database server. Raw encryption is not recommended when the data needs to be encrypted and decrypted only by the database server.

- **You cannot use TLS encryption** – You can use raw encryption instead of TLS encryption. Unlike TLS encryption, raw encryption cannot prevent replay or person-in-the-middle attacks, nor can it authenticate database servers.

**Example**

You need to send data from the binary_data column of the SensitiveData table in your database to a client that does not use databases. Because the data is sensitive, you encrypt the data into raw format using the following SQL statement:

```sql
SELECT ENCRYPT( binary_data, 'TheEncryptionKey', 'AES(FORMAT=RAW)', 'ThisIsTheIV' ) FROM SensitiveData;
```

You copy the encrypted data to the client along with an application that can decrypt the contents. You also provide the encryption key (TheEncryptionKey) and the initialization vector (ThisIsTheIV) to the client to use with the application. The client uses the application to decrypt the data and view it.

**Table encryption**

Table encryption allows you to encrypt tables or materialized views with sensitive data without the performance impact that encrypting the entire database might cause. When table encryption is enabled, table pages for the encrypted table, associated index pages, and temporary file pages are encrypted. The transaction log pages that contain transactions on encrypted tables are also encrypted.

To encrypt tables in your database, you must have table encryption enabled. Enabling table encryption must be done at database initialization. To see whether table encryption is enabled, query the EncryptionScope database property using the DB_PROPERTY function, as follows:

```sql
SELECT DB_PROPERTY( 'EncryptionScope' );
```

If the return value is TABLE, table encryption is enabled.

To see the encryption algorithm in effect for table encryption, query the Encryption database property using the DB_PROPERTY function, as follows:

```sql
SELECT DB_PROPERTY( 'Encryption' );
```
**Performance impact of table encryption**

For encrypted tables, each table page is encrypted when written to the disk, and is decrypted when read in from the disk. This process is invisible to applications. However, there may be a slight negative impact on performance when reading from, or writing to, encrypted tables. Encrypting or decrypting existing tables can take a long time, depending on the size of the table.

Index pages for indexes on columns in an encrypted table are also encrypted, as are transaction log pages containing transactions on the encrypted table, and all pages in the temporary file for the database. All other database and transaction log pages are unencrypted.

Encrypted tables can contain compressed columns. In this case, the data is compressed before it is encrypted.

Encrypting tables does not impact storage requirements.

**Starting a database that has table encryption enabled**

Starting a database that has table encryption enabled is the same as starting an encrypted database. For example, if the database is started with the -ek option, a key must be specified. If the database is started with the -ep option, you are prompted for the key.

**Enabling table encryption in a database (SQL)**

Create a database with table encryption by using the CREATE DATABASE statement, or enable table encryption in an existing database by using the CREATE ENCRYPTED TABLE DATABASE statement.

**Prerequisites**

By default, you must have the SERVER OPERATOR system privilege to execute the CREATE DATABASE statement and the CREATE ENCRYPTED TABLE DATABASE statement. The required privileges can be changed by using the -gu database server option.

**Task**

Table encryption must be enabled and configured at database creation time. If your database does not have table encryption enabled, or if you have database encryption in effect, using the CREATE ENCRYPTED TABLE DATABASE statement creates a copy of the database with table encryption enabled, and does not overwrite the original database file.

Create a database with table encryption, or enable table encryption on an existing database.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a database with table encryption</td>
<td>Create a database with the CREATE DATABASE statement, and specify a key and an encryption algorithm.</td>
</tr>
<tr>
<td>Option</td>
<td>Action</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Enable table encryption for an existing database</td>
<td>Create a copy of the database with the CREATE ENCRYPTED TABLE DATABASE statement, and specify a key.</td>
</tr>
</tbody>
</table>

Table encryption is enabled.

**Next**

You create an encrypted table by using the CREATE TABLE statement, or by altering an existing table to be encrypted by using the ALTER TABLE statement. When you encrypt a table, the key and/or algorithm specified when enabling table encryption is used.

*Enabling table encryption in a database (iqinit utility)*

You can enable table encryption during the creation of a database, using the command line.

**Prerequisites**

Table encryption must be enabled and configured at database creation time. You must re-create the database with table encryption enabled if your database does not have table encryption enabled, or if you have database encryption in effect.

**Task**

Create a database with the iqinit -et and -ek options, and specify a key and an encryption algorithm.

Table encryption is enabled.

*Encrypting a table*

You can create an encrypted table using the CREATE TABLE statement, or encrypt an existing table using the ALTER TABLE statement.

**Prerequisites**

To use the CREATE TABLE statement, you must have one of the following system privileges:

- CREATE TABLE
- CREATE ANY TABLE
- CREATE ANY OBJECT

To use the ALTER TABLE statement, you must be the owner of the table being altered or have one of the following privileges:

- ALTER privilege on the table
- ALTER ANY TABLE
ALTER ANY OBJECT

To encrypt tables in your database, table encryption must already be enabled in the database.

Task

When you encrypt a table, the encryption algorithm and key that were specified at database creation time are used.

You can either create a table with encryption, or encrypt an existing table.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a table with encryption</td>
<td>Create a table using the ENCRYPTED clause of the CREATE TABLE statement.</td>
</tr>
<tr>
<td>Encrypt an existing table</td>
<td>Encrypt a table with the ENCRYPTED clause of the ALTER TABLE statement.</td>
</tr>
</tbody>
</table>

The table is encrypted.

IPv6 Support

SAP Sybase 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 Sybase IQ supports both IPv4 and IPv6 addresses anywhere you can specify an IP address on the client or server.

ODBC classes support the use of IPv6 addresses for remote data access. JDBC classes do not support the use of IPv6 addresses for remote data access.

Setting up transport-layer security

The following steps provide an overview of the tasks required to set up transport-layer security.

1. 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). SAP Sybase IQ also provides functionality to create certificates, which is especially useful for development and testing.

2. If you are setting up transport-layer security for SAP Sybase IQ client/server applications:
   - **Start the SAP Sybase 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 do not use the TLS protocol. To prevent unencrypted connections from using the TDS protocol, specify the tcpip option -x tcpip(TDS=NO).

- **Configure client applications to use transport-layer security** – Specify the path and file name of trusted certificates using the Encryption connection parameter [ENC].

3. If you are setting up transport-layer security for SAP Sybase IQ web services:
   - **Start the SAP Sybase 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.
   - **Configure browsers or other web clients to trust certificates** – Encrypt SAP Sybase IQ web services.

4. If you are setting up an SAP Sybase 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.

---

**Digital certificates**

You need digital certificates to set up transport-layer security. You can obtain certificates from a certificate authority, or you can create them using the Certificate Creation utility (createcert).

*Certificate Creation utility*
You can use the Certificate Creation utility (createcert), to generate X.509 certificate files using RSA.

*Certificate Viewer utility*
You can use the Certificate Viewer utility, viewcert, to read X.509 certificates using RSA.

*Certificates for server authentication*
You can follow the same process to create certificate files for server authentication. In each case, you 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 or enterprise Certificate Authority.

- **Self-signed certificates** – Self-signed server certificates can be used for simple setups.
- **Enterprise root certificates** – An enterprise 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 an enterprise root certificate. Commercial Certificate Authorities have dedicated facilities to store private keys and create high-quality server certificates.

**Self-signed root certificates**

Self-signed root certificates can be used for simple setups involving a single database server.

**Tip:** Use enterprise level 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.

**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 Certificate Authority or an enterprise root certificate to sign identities.

**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 an enterprise root certificate or Certificate Authority. If you add a new database server, clients do not require a copy of the new certificate.
- **Security** – The enterprise 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 enterprise root certificate architecture.
Using certificates in a multi-server environment
To create certificates used in a multi-server environment:

- Generate a public enterprise root certificate and enterprise private key.
  Store the enterprise private key in a secure location, preferably a dedicated facility.
  For server authentication, you distribute the public enterprise root certificate to clients.
- Use the enterprise root certificate to sign identities.
  Use the public enterprise root certificate and enterprise 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.

**Enterprise root certificates**
Enterprise 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 server authentication, you can add servers without reconfiguring clients.

To set up enterprise root certificates, you create the enterprise root certificate and the enterprise private key that you use to sign identities.
Signed identity files
You can use an enterprise root certificate to sign server identity files.

For server authentication, you generate identity files for each server. Since these certificates are signed by an enterprise root certificate, you use the createcert -s option.

Globally-signed certificates
A commercial Certificate Authority is an organization that is in the business of creating high-quality certificates and using these certificates to sign your certificate requests.

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 an enterprise root certificate. This is only applicable if your Certificate Authority generates certificates that can be used to sign other certificates.
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:

You reference the server identity file and the password for the private key on the iqsvr16 command line.

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.

Note: When using a globally signed certificate, each client must verify field values to avoid trusting certificates that the same Certificate Authority has signed for other clients.
Utility Database Server Security

SAP Sybase IQ includes a phantom database, called the *utility database*, that has no physical representation.

There is no database file for this database, and the database can contain no data. The utility database can run on any SAP Sybase IQ server. In Sybase Control Center, the server for the utility database is known as the Utility Server.

The utility database permits a narrow range of specialized functions. It is provided so that you can 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.

**Defining the Utility Database Name When Connecting**

Specify the database name to start the utility database.

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. Specify `utility_db` as the database name when connecting.

For example:

```
dbisqlc -c "uid=dba;pwd=sql;eng=myserver;dbn=utility_db"
```

**Note:** When you connect to the utility database to create an IQ database having Windows raw partitions, note that 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 would be specified “\\\I:”. The backslash character is treated as an escape character in IQ databases but as a normal character in the utility database.

**Defining the Utility Database Password**

Define the user ID *DBA* for the utility database.

1. Use a text editor to open the file `util_db.ini`, which is stored in the server executable directory.
Because this directory is on the server, you can control access to the file, and thereby control who has access to the password.

2. In this line, replace "password" with the password you want to use:

```
[UTILITY_DB]
PWD=password
```

Use of the utility_db security level relies on the physical security of the computer hosting the database server, since the `util_db.ini` file can be easily read using a text editor.

**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.

**Table 1. Permissions for Role Administration**

<table>
<thead>
<tr>
<th><code>-gu</code> Switch Value</th>
<th>Effect</th>
<th>Applies To</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>Anyone can execute file administration statements</td>
<td>Any database including the utility database</td>
</tr>
<tr>
<td>none</td>
<td>No one can execute file administration statements</td>
<td>Any database including the utility database</td>
</tr>
<tr>
<td>DBA</td>
<td>Only users with the SERVER OPERATOR system privilege can execute file administration statements</td>
<td>Any database including the utility database</td>
</tr>
<tr>
<td>utility_db</td>
<td>Only the users who can connect to the utility database can execute file administration statements</td>
<td>Only the utility database</td>
</tr>
</tbody>
</table>

**Examples**

On 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:

```bash
start_iq -n testsrv -gu utility_db
```

On 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:

```bash
start_iq -n testsrv -gu utility_db -iqmt 256
```
Assuming that the utility database password was set to IQ&Mine49 during installation, this 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.

---

### Data Security

Since databases may contain proprietary, confidential, or private information, ensuring that the database and the data in it are designed for security is very important.

### System Secure Features

System secure features are features that you can make inaccessible to databases running on a database server.

When a feature is secured (made inaccessible), it is unavailable for use by client applications, database-defined stored procedures, triggers, and events. Secure feature settings apply to all databases running on the database server. Secure features are useful when you need to start a database that could contain embedded logic that you are uncertain of, such as a virus, or when you want to lock down the database server in situations where the database server or the database is 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 -sk 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 have created a system secure feature key, you can create customized secure feature keys that are assigned to a specific users, limiting users' access to only the features secured by the administrator for that key.

**Note:** The system secure feature key cannot be dropped unless a customized secure feature key has been created that has both the manage_features and manage_keys secure features enabled.

Customized secure feature keys are managed by select system procedures.
Creating secure feature keys

To control the database features available to users, use the secure features database server option (-sf) to specify the features that users are prevented from accessing on the database server.

Prerequisites

- Requires the SERVER OPERATOR system privilege
- Access to the manage_keys feature.

Task

Use the -sk database server option to create a system secure feature key. Use the sp_create_secure_feature_key system procedure to create a customized secure feature key.

Secure 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 secure feature key that can be used to manage access to secure features for a database server. If you want to alter the list of secured features once the database server is running, use the sa_server_option system procedure. To alter a customized secure feature key once the database server is running, use the sp_altern_secure_feature_key system 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.
   
   ```
   dbsrv16 -n secure_server -sf all -sk someSystemKey c:\mydata.db
   ```
   
2. Connect to the database server:
   
   ```
   dbisql -c
   "UID=DBA;PWD=sql;Host=myhost;Server=secure_server;DBN=demo"
   ```
   
3. Call the sp_use_secure_feature_key system procedure to specify that the secure feature key for the connection is the same as that specified by the -sk option:
   
   ```
   CALL sp_use_secure_feature_key ( 'system', 'someSystemKey' );
   ```
   
4. To change the secure features of the system secure feature key, use the sa_server_option system procedure. For example:
   
   ```
   CALL sa_server_option( 'SecureFeatures', '-remote_data_access' );
   ```
5. Create a customized secure feature key for the user Bob, that allows Bob to send emails:

```sql
CALL sp_create_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' , 'sa_send_email' );
```

6. After logging into the database, Bob must run the following command to send emails:

```sql
CALL sp_use_secure_feature_key ( 'bobsKey' , 'anotherAuthKey' );
```

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 sa_send_email feature.

**See also**

- `-sk iqsrv16 database server option on page 305`
- `-sf iqsrv16 database server option on page 306`
- `sp_alter_secure_feature_key System Procedure on page 366`
- `sp_create_secure_feature_key System Procedure on page 316`
- `sp_drop_secure_feature_key System Procedure on page 367`
- `sp_list_secure_feature_keys System Procedure on page 368`
- `sp_use_secure_feature_key System Procedure on page 368`
External Authentication

SAP Sybase IQ supports LDAP and Kerberos external authentication methods.

LDAP User Authentication with SAP Sybase IQ

You can integrate SAP Sybase IQ into any existing enterprise-wide directory access framework based on Lightweight Directory Access Protocol (LDAP), a widely accepted international standard.

Integration of SAP Sybase 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

License Requirements for LDAP User Authentication

The Advanced Security Option (IQ_SECURITY) protects your environment against unauthorized access, and is required to allow LDAP user authentication with SAP Sybase IQ.

LDAP Server Configuration Object

SAP Sybase 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 Sybase 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 Sybase 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 Sybase IQ internal configuration object. LDAP server refers to the external entity.
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 is cannot connect to the LDAP server, the designated secondary LDAP server configuration object is used for user authentication. You can manually manage fail over and fail back using with SQL statements or be performed automatically by SAP Sybase 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 Sybase IQ will be unavailable.

Workflow to Enable LDAP User Authentication

There is distinct workflow to enabling LDAP user authentication with SAP Sybase IQ.

1. Add LDAP User Authentication as a Login Method.
2. Create an LDAP Server Configuration Object.
3. Validate the LDAP Server Configuration Object.
4. Define the LDAP user authentication login policy options in any login policy (including root) assigned to any user using LDAP user authentication.
5. Assign an LDAP user authentication enabled login policy to an existing user to use LDAP user authentication.
6. Once configuration is complete, execute the sa_get_ldapserver_status stored procedure to verify that each LDAP server database connection is in a READY or ACTIVE state.
7. Verify that users can log in using LDAP user authentication

Manage LDAP User Authentication with SAP Sybase IQ

Management includes the creation, modification and option maintenance of the LDAP server configuration object to facilitate LDAP user authentication.
Adding 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.

Task

Once set, LDAP user authentication is immediately available.

To add the LDAPUA value to the LOGIN_MODE option, execute:

```
SET OPTION PUBLIC.login_mode = LDAPUA
```

See also

- `LOGIN_MODE Option` on page 297

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.

If LDAP user authentication is the only authentication method allowed to access the SAP Sybase IQ database, these circumstances may create a scenario in which no user is permitted to log on:

- Of 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 Sybase 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 need not 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.

To allow standard authentication, execute one of these commands:

<table>
<thead>
<tr>
<th>Level</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>start_iq -al &quot;user1,user2,user3&quot; server_name.cfg database-name.db</td>
</tr>
<tr>
<td>Database</td>
<td>start_iq servername.cfg database_name.db -al &quot;user1,user2,user3&quot;</td>
</tr>
</tbody>
</table>

**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
```

**See also**

- -al iqsrv16 Server Option on page 299
- -al iqsrv16 Database Option on page 299

**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.

**Task**

During LDAP user authentication, SAP Sybase 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 Sybase applications on that machine.

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:sybase\shared, in a file called \trusted.txt:

```
SET OPTION PUBLIC.TRUSTED_CERTIFICATES_FILE = 'C:sybase\shared\trusted.txt'
```

See also
- `TRUSTED_CERTIFICATES_FILE Option` on page 298

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.

Task
The LDAP server configuration object provides a connection between SAP Sybase IQ and a physical LDAP serve. 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.

1. Identify the values for the applicable SEARCH DN attributes to be defined for the new LDAP server configuration object.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> See <code>Syntax and Parameters for the LDAP Server Configuration Object URL</code> for supported syntax.</td>
</tr>
<tr>
<td>ACCESS ACCOUNT</td>
<td>The distinguished name for a user connecting to the external LDAP server.</td>
</tr>
<tr>
<td>IDENTIFIED BY</td>
<td>The password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
<tr>
<td>IDENTIFIED BY ENCRYPTED</td>
<td>The encrypted password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
</tbody>
</table>

2. Identify the values for the applicable LDAPUA server attributes for the new LDAP server configuration object.
Table 3. LDAPUA Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH DN</td>
<td>All attributes defined from SEARCH DN Attributes (see step 1).</td>
</tr>
<tr>
<td>AUTHENTICATION URL</td>
<td>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.</td>
</tr>
<tr>
<td>CONNECTION TIMEOUT</td>
<td>Specifies the connection timeout value for both DN searches and authentication between SAP Sybase IQ and the external LDAP server. Specified in milliseconds, the default value is 10 seconds.</td>
</tr>
<tr>
<td>CONNECTION RETRIES</td>
<td>Specifies the number of retries on connections from SAP Sybase 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.</td>
</tr>
<tr>
<td>TLS</td>
<td>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.</td>
</tr>
</tbody>
</table>

3. Execute the CREATE LDAP SERVER command, 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
```

See also

- Syntax and Parameters for the LDAP Server Configuration Object URL on page 176
- Enabling Secure LDAP on page 176
- CREATE LDAP SERVER Statement on page 245
- Editing LDAP Server Configuration Object Attributes on page 171
- Setting the TLS Connection Trusted Relationship on page 166
Validating an LDAP Server Configuration Object

Validate changes to the attribute of an existing LDAP server configuration object.

Prerequisites

Requires the MANAGE ANY LDAP SERVER system privilege.

Task

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 Sybase 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 use the userID with the search 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.

1. Identify the SEARCH DN attributes of the LDAP server configuration object to be validated.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> See <em>Syntax and Parameters for the LDAP Server Configuration Object URL</em> for supported syntax.</td>
</tr>
<tr>
<td>ACCESS ACCOUNT</td>
<td>The distinguished name for a user connecting to the external LDAP server.</td>
</tr>
<tr>
<td>IDENTIFIED BY</td>
<td>The password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
<tr>
<td>IDENTIFIED BY ENCRYPTED</td>
<td>The encrypted password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
</tbody>
</table>

2. Identify the LDAPUA attributes of the LDAP server configuration object to be validated.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH DN</td>
<td>All attributes defined from SEARCH DN Attributes (see step 1).</td>
</tr>
</tbody>
</table>
3. Execute the **VALIDATE LDAP SERVER** command with the applicable attributes.

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'
```

**See also**
- *Enabling Secure LDAP* on page 176
- *Syntax and Parameters for the LDAP Server Configuration Object URL* on page 176
- **VALIDATE LDAP SERVER Statement** on page 294
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
Requires the MANAGE ANY LDAP SERVER system privilege.

Task
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 Sybase IQ server. Upon successful authentication of a user, the connection state to the LDAP server changes to ACTIVE.
To activate an LDAP server configuration object, execute:

```
ALTER LDAP SERVER LDAP_server_name WITH ACTIVATE
```

See also
• ALTER LDAP SERVER Statement on page 231
• LDAP Server Configuration Object States on page 175

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.

Task
1. Identify the existing SEARCH DN attributes to be modified.
### Table 6. SEARCH DN Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> See <em>Syntax and Parameters for the LDAP Server Configuration Object URL</em> for supported syntax.</td>
</tr>
<tr>
<td>ACCESS ACCOUNT</td>
<td>The distinguished name for a user connecting to the external LDAP server.</td>
</tr>
<tr>
<td>IDENTIFIED BY</td>
<td>The password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
<tr>
<td>IDENTIFIED BY ENCRYPTED</td>
<td>The encrypted password associated with the ACCESS ACCOUNT distinguished name.</td>
</tr>
</tbody>
</table>

2. Identify the existing LDAPUA attributes to be modified.

### Table 7. LDAPUA Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEARCH DN</td>
<td>All attributes defined from SEARCH DN Attributes (see step 1).</td>
</tr>
<tr>
<td>AUTHENTICATION URL</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> See <em>Syntax and Parameters for the LDAP Server Configuration Object URL</em> for supported syntax.</td>
</tr>
<tr>
<td>CONNECTION TIMEOUT</td>
<td>Specifies the connection timeout value for both DN searches and authentication between SAP Sybase IQ and the external LDAP server. Specified in milliseconds, the default value is 10 seconds.</td>
</tr>
<tr>
<td>CONNECTION RETRIES</td>
<td>Specifies the number of retries on connections from SAP Sybase 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.</td>
</tr>
<tr>
<td>TLS</td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> See <em>Enabling Secure LDAP and Setting the TLS Connection Trusted Relationship</em>.</td>
</tr>
</tbody>
</table>
3. Identify the server clauses to be used.

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WITH SUSPEND</td>
<td>Puts the LDAP server into maintenance mode</td>
</tr>
<tr>
<td>WITH ACTIVATE</td>
<td>Puts the LDAP server in a READY state and enables LDAP authentication</td>
</tr>
<tr>
<td>WITH REFRESH</td>
<td>Reinitializes LDAP user authentication</td>
</tr>
</tbody>
</table>

4. Execute the `ALTER LDAP SERVER` command with the applicable parameters and clauses, for example:

```
ALTER LDAP SERVER apps_primary
AUTHENTICATION URL 'ldap://my_LDAPserver:1066/
CONNECTION RETRIES 10
WITH ACTIVATE
```

See also
- Syntax and Parameters for the LDAP Server Configuration Object URL on page 176
- Enabling Secure LDAP on page 176
- `ALTER LDAP SERVER` Statement on page 231
- Setting the TLS Connection Trusted Relationship on page 166
- Validating an LDAP Server Configuration Object on page 169

**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.

**Task**
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 Sybase 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 Sybase 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.

To refresh the LDAP server, execute:
ALTER LDAP SERVER  LDAP_server_name
WITH REFRESH

See also
•  ALTER LDAP SERVER Statement on page 231
•  LDAP Server Configuration Object States on page 175

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.

Task
To suspend an LDAP server, execute:

ALTER LDAP SERVER  LDAP_server_name
WITH SUSPEND

See also
•  ALTER LDAP SERVER Statement on page 231
•  LDAP Server Configuration Object States on page 175

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.

Task
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
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
```

See also

- *DROP LDAP SERVER Statement* on page 257
- *LDAP Server Configuration Object States* on page 175

### 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.
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 Sybase IQ.

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]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>The host name of the LDAP server.</td>
</tr>
<tr>
<td>port</td>
<td>The port number of the LDAP server.</td>
</tr>
<tr>
<td>node</td>
<td>The node in the object hierarchy at which to start the search.</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes</td>
<td>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.</td>
</tr>
<tr>
<td>base</td>
<td>Qualifies the search criteria.</td>
</tr>
<tr>
<td>one</td>
<td>Specifies a search of the base node.</td>
</tr>
<tr>
<td>sub</td>
<td>Specifies a search of node and all sublevels.</td>
</tr>
<tr>
<td>filter</td>
<td>Specifies the attribute or attributes used to search for a database user's distinguished name (DN). The filter can be simple, such as &quot;uid=<em>,&quot; or compound, such as &quot;(uid=</em>)(ou=group).&quot; 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.</td>
</tr>
</tbody>
</table>

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. There are no default values for these parameters. 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 Sybase IQ.

### LDAP User Authentication Login Policy Options

There are several login policy options that are specific to LDAP user authentication. You must define these options in any login policy (including root) assigned to any user using LDAP user authentication.

You can define the options that are specific to LDAP server database objects when initially creating a login policy, or you can add them to existing policies, including the root login policy. The MANAGE ANY LOGIN POLICY system privilege is required to set these login policy options.

**Modifying the Root Login Policy**

You can modify the option values for the root login policy, but you cannot drop the policy.

**Prerequisites**

Requires the MANAGE ANY LOGIN POLICY system privilege.
Task
Each new database is created with a default login policy, called the root policy. When you create a user account without specifying a login policy, the user becomes part of the root login policy.
To modify the options of the root login policy, execute:

```
ALTER LOGIN POLICY ROOT {login_policy_options}
```

See also
- LDAP User Authentication Login Policy Options on page 177
- ALTER LOGIN POLICY Statement on page 233

Modifying an Existing Login Policy
Use Interactive SQL to change the options for an existing login policy.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
To alter the options of an existing login policy, execute:

```
ALTER LOGIN POLICY policy-name {login_policy_options}
```

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

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

See also
- LDAP User Authentication Login Policy Options on page 177
- ALTER LOGIN POLICY Statement on page 233

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.

Task
Login policy names must be unique. An error message appears if the login policy name already exists.
To create a new login policy, 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

See also
• LDAP User Authentication Login Policy Options on page 177
• CREATE LOGIN POLICY Statement on page 248

Assigning a Login Policy to an Existing User
Use Interactive SQL to assign a login policy to an existing user.

Prerequisites
Requires the MANAGE ANY LOGIN POLICY system privilege.

Task
1. To change the login policy assigned to a user, execute:
   ALTER USER userID
   LOGIN POLICY policy_name
2. Have the user log out and back in to apply the new login policy.

See also
• ALTER USER Statement on page 242

Manage Users and Passwords with LDAP User Authentication
To log in to SAP Sybase IQ using LDAP user authentication, each user must have an active user ID and password on both the external LDAP server and the SAP Sybase IQ server.

Define passwords when creating users in SAP Sybase IQ. Thereafter, however, maintain any password changes for users configured to use LDAP user authentication on the external LDAP server, not on the SAP Sybase IQ server. The first time a user logs in to SAP Sybase IQ using LDAP user authentication after a password change, the password in the SAP Sybase IQ database for the user account is automatically updated with the new (LDAP defined) password.

If a password is updated in SAP Sybase IQ, the next time the user authenticates using LDAP user authentication, the password in the SAP Sybase IQ database is overwritten with the password from the LDAP server.

Since passwords are automatically synchronized between the LDAP and SAP Sybase IQ servers, if a user is granted the ability to log on using Standard authentication (the password...
defined in the SAP Sybase IQ database), he or she must still use the LDAP server credentials to authenticate.

**Display Current Status Information for a User**

The *sa_get_user_status* stored procedure generates 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.

**See also**

- *sa_get_user_status system procedure* on page 314

**Display Current Status for an LDAP Server Configuration Object**

The *sa_get_ldapserver_status* stored procedure generates a report on the current status 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.

No system privileges are required to run this stored procedure.

**See also**

- *sa_get_ldapserver_status System Procedure* on page 313

**Kerberos 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.

**Warning!** There are important security implications to consider when using Kerberos logins as a single security solution.

SAP Sybase 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 SAP Sybase IQ clients in the Kerberos connection parameter. SAP Sybase IQ database servers cannot use SSPI—they need 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.

SAP Sybase IQ supports Kerberos authentication from DBLib, ODBC, OLE DB, and ADO.NET clients, and Sybase Open Client and jConnect clients. Kerberos authentication can be used with SAP Sybase IQ transport layer security encryption, but SAP Sybase IQ does not support Kerberos encryption for network communications.

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.

**Kerberos clients**

Kerberos authentication is available on several platforms. For a list of tested Kerberos clients, see [http://www.sybase.com/detail?id=1061807](http://www.sybase.com/detail?id=1061807).

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 SAP Sybase IQ clients in the Kerberos connection parameter. SAP Sybase IQ database servers cannot use SSPI—they need a supported Kerberos client other than SSPI.

<table>
<thead>
<tr>
<th>Kerberos client</th>
<th>Default keytab file</th>
<th>GSS-API library file name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows MIT Kerberos client</td>
<td>C:\WINDOWS \krb5kt</td>
<td>gssapi32.dll or gssapi64.dll</td>
<td>The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.</td>
</tr>
<tr>
<td>Windows CyberSafe Kerberos client</td>
<td>C:\Program Files\CyberSafe\v5srvtab</td>
<td>gssapi32.dll or gssapi64.dll</td>
<td>The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.</td>
</tr>
<tr>
<td>Unix MIT Kerberos client</td>
<td>/etc/krb5.keytab</td>
<td>libgssapi_krb5.so (^1)</td>
<td>The KRB5_KTNAME environment variable can be set before starting the database server to specify a different keytab file.</td>
</tr>
<tr>
<td>Unix CyberSafe Kerberos client</td>
<td>/krb5/v5srvtab</td>
<td>libgss.so (^1)</td>
<td>The CSFC5KTNAME environment variable can be set before starting the database server to specify a different keytab file.</td>
</tr>
<tr>
<td>Unix Heimdal Kerberos client</td>
<td>/etc/krb5.keytab</td>
<td>libgssapi.so.1 (^1)</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) These file names may vary depending on your operating system and Kerberos client version.

### Setting up a Kerberos system to use with SAP Sybase IQ

You can configure Kerberos authentication to be used with SAP Sybase IQ.

#### Prerequisites

You must be logged in to your computer using Kerberos authentication.
Task

Kerberos is a network authentication protocol that provides strong authentication and encryption using secret-key cryptography.

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 Sybase IQ database server.

   The default Kerberos principal for the database server has the format server-name@REALM, where server-name is the SAP Sybase IQ database server name. To use a different server principal, use the -kp server option. Principals are case significant, 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 principal server-name@REALM from the KDC to the computer running the SAP Sybase 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 Sybase IQ server can read it, but unauthorized users do not have read permission.

   The Kerberos system is authenticated and configured to be used with SAP Sybase IQ.

Next

Configure your SAP Sybase IQ database server and database to use Kerberos.

Configuring SAP Sybase IQ databases to use Kerberos

You can configure SAP Sybase IQ 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 Sybase IQ can use it.

**Task**

The Kerberos login feature allows you to maintain a single user ID and password for database connections, operating systems, and network logins.

1. Start the SAP Sybase IQ 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:
   
   ```sql
   SET OPTION PUBLIC.login_mode = 'Kerberos,Standard';
   ```

   **Warning!** 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 system 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:
   
   ```sql
   CREATE USER "kerberos-user"
   IDENTIFIED BY abcd123;
   ```

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:
   
   ```sql
   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:
   
   ```bash
   dbisql -c "KERBEROS=YES;Server=my_server_princ"
   ```

The SAP Sybase IQ database is configured to use Kerberos authentication.
Next

You can use Kerberos authentication to connect from a client. Optionally, you can create a Kerberos login mapping.

**Connections from a Sybase Open Client or a jConnect application**

To connect from a Sybase Open Client or jConnect application:

- Set up Kerberos authentication.
- Configure SAP Sybase IQ to use Kerberos.
- Set up Sybase Open Client or jConnect as you would for Kerberos authentication with Adaptive Server Enterprise. The server name must be the SAP Sybase IQ server's name and is case significant. You cannot connect using an alternate server name from Sybase Open Client or jConnect.

**Using SSPI for Kerberos logins on Windows**

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.

**Prerequisites**

You must already have Kerberos configured before SAP Sybase IQ can use it. You must already have your SAP Sybase IQ database server and database configured to use Kerberos.

**Task**

SSPI can only be used by SAP Sybase IQ clients in the Kerberos connection parameter. SAP Sybase IQ database servers cannot use SSPI—they need a supported Kerberos client other than SSPI.

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';
```

You can use SSPI for Kerberos authentication on Windows.
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**

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Common solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Unable to load Kerberos GSS-API library&quot; message</td>
<td>• Ensure a Kerberos client is installed on the database server computer, including the GSS-API library.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• Ensure the directory and any supporting libraries is listed in the library path (%PATH% on Windows).</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>Symptom</td>
<td>Common solutions</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
</tr>
</tbody>
</table>
| "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 Sybase 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:
<table>
<thead>
<tr>
<th>Symptom</th>
<th>Common solutions</th>
</tr>
</thead>
</table>
| "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 by SAP Sybase IQ on both the client and server platforms. |
| "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. |
### Symptom

"The login ID 'client-Kerberos-principal' has not been mapped to any database user ID"

### Common solutions

- The Kerberos principal must be mapped to a database user ID using the GRANT KERBEROS LOGIN statement. Note 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.

---

#### Security concerns: Temporary public options for added security

Setting the value of the login_mode option for a given database to allow a combination of Standard, Integrated, Kerberos, and LDAPUA logins using the SET OPTION statement permanently enables the specified types of logins for that database. For example, the following statement permanently enables standard and integrated logins:

```sql
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:

```sql
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. When you add integrated, Kerberos, or LDAPUA logins to your database, the database relies on the security of the operating system on which it is running. If the database is copied to another computer, access to the database reverts to the SAP Sybase IQ security model.
Security concerns: Copied database files

If the database file can be copied, use the temporary public login_mode option for integrated and Kerberos logins. If the file is copied, the integrated and Kerberos logins are not supported 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.
• Set the 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.
• Strongly encrypt the database file using the AES encryption algorithm. The encryption key should be complex and difficult to guess.

Licensing Requirements for Kerberos

The Advanced Security Option (IQ_SECURITY) protects your environment against unauthorized access, and is required to use Kerberos authentication with SAP Sybase IQ.
Advanced Security Options in SAP Sybase IQ

The SAP® Sybase® 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 Sybase IQ option.

FIPS Support in SAP Sybase IQ

SAP Sybase IQ supports Federal Information Processing Standards (FIPS)-approved encryption technology. FIPS is supported on all platforms supported by SAP Sybase IQ.

The main impact of FIPS support for SAP Sybase IQ is that encryption can be nondeterministic, which is the default behavior. A nondeterministic algorithm is one in which the same input yields different output values each time. This means that when you use a key to encrypt a string, the encrypted string is different each time. The algorithm, however, can still decrypt the nondeterministic result using the key. This feature makes analyzing the encryption algorithm more difficult, and encryption more secure.

Support of FIPS is part of the separately licensed SAP Sybase IQ Advanced Security Option.

Both RSA and FIPS security are included with SAP Sybase IQ. RSA encryption requires no separate libraries, but FIPS requires two optional libraries: dbfips11.dll and sbgse2.dll. The library sbgse2.dll is provided by Certicom. Both security models require certificates. The rsaserver certificate is named rsaserver.id.

FIPS also requires this registry setting, which is set automatically by the SAP Sybase IQ installation utility:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Certicom\libsb]
"expectedtag"=hex:5b,0f,4f,a6,e2,4a,ef,3b,
44,07,05,2e,b0,49,02,71,1f,d9,91,b6
```

Licensing Requirements for FIPS Support

The Advanced Security Option (IQ_SECURITY) is required to use FIPS authentication with SAP Sybase IQ.

FIPS-certified encryption technology

You can use FIPS-certified security algorithms to encrypt your database files, or to encrypt communications for database client/server communication, web services, and .

Federal Information Processing Standard (FIPS) 140-2 specifies requirements for security algorithms. FIPS 140-2 is granted by the American and Canadian governments through the
National Institute of Standards and Testing (NIST) and the Canadian Communications Security Establishment (CSE).

SAP Sybase IQ uses a FIPS-certified module for encryption from Certicom. On Windows (desktop and Windows Mobile) and Unix platforms, SAP Sybase IQ uses Certicom Security Builder GSE (FIPS Module v2.0). This is number 542 on the page http://csrc.nist.gov/cryptval/140-1/140val-all.htm.

Enforcing FIPS
Optionally, you can enforce the use of FIPS-certified encryption on the client or server with a FIPS option. When you set the FIPS option to on, all secure communications must be FIPS-certified. If someone tries to use non-FIPS RSA encryption, it is automatically upgraded to FIPS-certified RSA encryption. The FIPS option can be set on the client or server on which you want FIPS-certified encryption to be enforced. SAP Sybase IQ has a -fips command line option, and clients have a fips option that can be set with the encryption connection parameter.

Column Encryption in SAP Sybase IQ

SAP Sybase IQ supports user-encrypted columns.

Strong encryption of the SAP Sybase IQ database file uses a 128-bit algorithm and a security key. The data is unreadable and virtually undecipherable without the key. The algorithm supported is described in FIPS-197, the Federal Information Processing Standard for the Advanced Encryption Standard.

SAP Sybase 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.

Certain database options affect column encryption.

See also
• Database Options for Column Encryption on page 219

Licensing Requirements for Column Encryption

The Advanced Security Option (IQ_SECURITY) is required to use user-encrypted columns with SAP Sybase IQ.

Definitions of Encryption Terms
Definitions of terms used when describing encryption of stored data.

• 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.
• ciphertext – data in an unintelligible form that preserves the information content of the plaintext form.
• encryption – a reversible transformation of data from plaintext to ciphertext. Also known as enciphering.
• decryption – the reverse transformation of ciphertext back to plaintext. Also known as deciphering.
• 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 Sybase IQ interfaces accept character strings as keys.
• 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.
• 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 Sybase IQ.

Data Types for Encrypted Columns
The data types supported for encrypted columns and working with these data types.

Supported Data Types
The first parameter of the AES_ENCRYPT function must be one of the supported data types.

<table>
<thead>
<tr>
<th>CHAR</th>
<th>NUMERIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR</td>
<td>FLOAT</td>
</tr>
<tr>
<td>TINYINT</td>
<td>REAL</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>INTEGER</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>BIGINT</td>
<td>DATE</td>
</tr>
<tr>
<td>BIT</td>
<td>TIME</td>
</tr>
<tr>
<td>BINARY</td>
<td>DATETIME</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>UNSIGNED INT</td>
<td>SMALLDATETIME</td>
</tr>
<tr>
<td>UNSIGNED BIGINT</td>
<td></td>
</tr>
</tbody>
</table>

The LOB data type is not currently supported for SAP Sybase IQ column encryption.
**Preservation of Data Types**
SAP Sybase 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 Sybase 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`.

**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');
```

See also
• `AES_ENCRYPT Function [String]` on page 195

**AES_ENCRYPT Function [String]**

Encrypts the specified values using the supplied encryption key, and returns a `VARBINARY` or `LONG VARBINARY`.

**Syntax**

```
AES_ENCRYPT( string-expression, key )
```

**Parameters**

- `string-expression` – the data to be encrypted. You can also pass binary values to `AES_ENCRYPT`. This parameter is case-sensitive, even in case-insensitive databases.

- `key` – the encryption key used to encrypt the `string-expression`. To obtain the original value, also 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 is difficult to guess. Choose a value that is at least 16 characters long, contains a mix of uppercase and lowercase letters, and includes numbers and special characters. You need this key each time you want to decrypt the data.

**Warning!** Protect your key; store a copy of your key in a safe location. If you lose your key, encrypted data becomes completely inaccessible and unrecoverable.

**Usage**

`AES_ENCRYPT` returns a `VARBINARY` value, which is at most 31 bytes longer than the input `string-expression`. The value returned by this function is the ciphertext, which is not human-readable. You can use the `AES_DECRYPT` function to decrypt a `string-expression` that was encrypted with the `AES_ENCRYPT` function. To successfully decrypt a `string-expression`, use
the same encryption key and algorithm used to encrypt the data. If you specify an incorrect encryption key, an error is generated.

If you are storing encrypted values in a table, the column should be of data type VARBINARY or VARCHAR, and greater than or equal to 32 bytes, so that character set conversion is not performed on the data. (Character set conversion prevents data decryption.) If the length of the VARBINARY or VARCHAR column is fewer than 32 bytes, the AES_DECRYPT function returns an error.

The result data type of an AES_ENCRYPT function may be a LONG BINARY. If you use AES_ENCRYPT in a SELECT INTO statement, you must have an Unstructured Data Analytics Option license, or use CAST and set AES_ENCRYPT to the correct data type and size.

Standards and Compatibility

- SQL – vendor extension to ISO/ANSI SQL grammar.
- Sybase – not supported by Adaptive Server Enterprise.

See also

- AES_DECRYPT Function [String] on page 198
- Encryption and Decryption Example on page 221
- LOAD TABLE ENCRYPTED Clause on page 199
- Effect of Different Data Types on Ciphertext on page 194
- Data Types for Encrypted Columns on page 193

REPLACE Function [String]
Replaces all occurrences of a substring with another substring.

Syntax

```sql
REPLACE ( original-string, search-string, replace-string )
```

Parameters

If any argument is NULL, the function returns NULL.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>original-string</td>
<td>The string to be searched. This string can be any length.</td>
</tr>
<tr>
<td>search-string</td>
<td>The string to be searched for and replaced with replace-string. This string is limited to 255 bytes. If search-string is an empty string, the original string is returned unchanged.</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace-string</td>
<td>The replacement string, which replaces search-string. This can be any length. If replace-string is an empty string, all occurrences of search-string are deleted.</td>
</tr>
</tbody>
</table>

### Returns

**LONG VARCHAR**

**LONG NVARCHAR**

*Note:* The result data type is a **LONG VARCHAR**. If you use `REPLACE` in a **SELECT INTO** statement, you must have an Unstructured Data Analytics Option license or use `CAST` and set `REPLACE` to the correct data type and size.

### Examples

The following statement returns the value “xx.def.xx.ghi:”

```sql
SELECT REPLACE( 'abc.def.abc.ghi', 'abc', 'xx' ) FROM iq_dummy
```

The following statement generates a result set containing `ALTER PROCEDURE` statements which, when executed, repair stored procedures that reference a table that has been renamed. (To be useful, the table name must be unique.)

```sql
SELECT REPLACE( replace(proc_defn,'OldTableName','NewTableName'), 'create procedure', 'alter procedure') FROM SYS.SYSPROCEDURE WHERE proc_defn LIKE '%OldTableName%'
```

Use a separator other than the comma for the `LIST` function:

```sql
SELECT REPLACE( list( table_id ), ',', '--' ) FROM  SYS.ISYSTAB WHERE table_id <= 5
```

### Usage

The result data type of a `REPLACE` function is a **LONG VARCHAR**. If you use `REPLACE` in a **SELECT INTO** statement, you must have an Unstructured Data Analytics Option license, or use `CAST` and set `REPLACE` to the correct data type and size.

There are two ways to work around this issue:

- Declare a local temporary table, then perform an `INSERT`:

  ```sql
  DECLARE local temporary table #mytable
  (name_column char(10)) on commit preserve rows;
  INSERT INTO #mytable SELECT REPLACE(name,'0','1')   FROM dummy_table01;
  ```
Advanced Security Options in SAP Sybase IQ

- Use **CAST**:

```sql
SELECT CAST(replace(name, '0', '1') AS Char(10)) into #mytable from dummy_table01;
```

If you need to control the width of the resulting column when `replace-string` is wider than `search-string`, use the **CAST** function. For example,

```sql
CREATE TABLE aa(a CHAR(5));
INSERT INTO aa VALUES('CCCCC');
COMMIT;
SELECT a, CAST(REPLACE(a,'C','ZZ') AS CHAR(5)) FROM aa;
```

**Standards and Compatibility**

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Compatible with Adaptive Server Enterprise.

**AES_DECRYPT Function [String]**

Decrypts the string using the supplied key, and returns, by default, a **VARBINARY** or **LONG BINARY**, or the original plaintext type.

**Syntax**

```sql
AES_DECRYPT( string-expression, key [, data-type ] )
```

**Parameters**

- `string-expression`—the string to be decrypted. You can also pass binary values to this function. This parameter is case sensitive, even in case-insensitive databases.

- `key`—the encryption key required to decrypt the `string-expression`. To obtain the original value that was encrypted, the key must be the same encryption key that was used to encrypt the `string-expression`. This parameter is case-sensitive, even in case-insensitive databases.

**Warning!** Protect your key; store a copy of your key in a safe location. If you lose your key, the encrypted data becomes completely inaccessible and unrecoverable.

- `data-type`—this optional parameter specifies the data type of the decrypted `string-expression` and must be the same data type as the original plaintext.

If you do not use a **CAST** statement while inserting data using the **AES_ENCRYPT** function, you can view the same data using the **AES_DECRYPT** function by passing **VARCHAR** as the `data-type`. If you do not pass `data-type` to **AES_DECRYPT**, **VARBINARY** data type is returned.

**Usage**

You can use the **AES_DECRYPT** function to decrypt a `string-expression` that was encrypted with the **AES_ENCRYPT** function. This function returns a **VARBINARY** or **LONG VARBINARY** value with the same number of bytes as the input string, if no data type is specified. Otherwise, the specified data type is returned.
To successfully decrypt a *string-expression*, you must use the same encryption key that was used to encrypt the data. An incorrect encryption key returns an error.

**Example**

Decrypt the password of a user from the `user_info` table.

```sql
SELECT AES_DECRYPT(user_pwd, '8U3dkA', CHAR(100))
FROM user_info;
```

**Standards and Compatibility**

- SQL – vendor extension to ISO/ANSI SQL grammar.
- Sybase – not supported by Adaptive Server Enterprise.

**See also**

- *AES_ENCRYPT Function [String]* on page 195
- *Encryption and Decryption Example* on page 221
- *LOAD TABLE ENCRYPTED Clause* on page 199
- *Data Types for Encrypted Columns* on page 193

**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**

```sql
| ENCRYPTED(data-type 'key-string' [, 'algorithm-string' ] )
```

**Parameters**

- *data-type* – the data type that the input file field should be converted to as input to the `AES_ENCRYPT` 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.
Warning! 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, AES is the default, as it is the only supported algorithm. AES 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 **VARBINARY**. Specifying other data types returns an error.

### LOAD TABLE ENCRYPTED 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;
```

### See also
- **AES_ENCRYPT Function [String]** on page 195
- **AES_DECRYPT Function [String]** on page 198
- **Encryption and Decryption Example** on page 221
- **Data Types for Encrypted Columns** on page 193

### LOAD TABLE Statement
Imports data into a database table from an external file.

### Syntax
```
LOAD [ INTO ] TABLE [ owner.]table-name
... ( load-specification [ , ... ] )
```
Advanced Security Options in SAP Sybase IQ

Parameters

- **load-specification** —

  ```
  { column-name [ column-spec ]
    | FILLER ( filler-type )
  }
  ```

- **column-spec** —

  ```
  { ASCII ( input-width )
    | BINARY [ WITH NULL BYTE ]
    | PREFIX { 1 | 2 | 4 }
    | 'delimiter-string'
    | DATE [ FORMAT ] ( input-date-format ) [, input-date-format, ...]
    | DATETIME [ FORMAT ] ( input-datetime-format [, input-datetime-format, ...] )
    | ENCRYPTED ( data-type 'key-string' [, 'algorithm-string' ] )
    | DEFAULT default-value
  } [ NULL ( { BLANKS | ZEROS | 'literal', ...} ) ]
  ```

- **filler-type** —

  ```
  { input-width
    | PREFIX { 1 | 2 | 4 }
    | 'delimiter-string'
  }
  ```

- **constrainttype** —

  ```
  { CHECK integer
    | UNIQUE integer
  }
  ```
Examples

• Example 1 – Load data from one file into the Products table on a Windows system. A tab is used as the column delimiter following the Description and Color columns:

```sql
LOAD TABLE Products
(ID ASCII(6),
FILLER(1),
Name   ASCII(15),
FILLER(1),
Description   '\x09',
Size   ASCII(2),
FILLER(1),
Color   '\x09',
Quantity   PREFIX 2,
UnitPrice   PREFIX 2,
FILLER(2) )
FROM 'C:\mydata\source1.dmp'
QUOTES OFF
ESCAPES OFF
BYTE ORDER LOW
NOTIFY 1000
```

• Example 2 – Load data from a file a.inp on a client computer:

```sql
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT FILE 'c:\client-data\a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:\client-data\m.log'
ROW LOG 'c:\client-data\r.log'ONLY LOG UNIQUE
```

• Example 3 – Load data from two files into the product_new table (which allows NULL values) on a UNIX system. The tab character is the default column delimiter, and the newline character is the row delimiter:

```sql
LOAD TABLE product_new
(id,
name,
description,
...)
FROM 'source1'
FROM 'source2'
```

Advanced Security Options in SAP Sybase IQ
• **Example 4** – Ignore 10 word-length violations; on the 11th, deploy the new error and roll back the load:

```sql
load table PTAB1(
    ck1         ','  null ('NULL') ,
    ck3fk2c2    ','  null ('NULL') ,
    ck4         ','  null ('NULL') ,
    ck5         ','  null ('NULL') ,
    ck6c1       ','  null ('NULL') ,
    ck6c2       ','  null ('NULL') ,
    rid         ','  null ('NULL')  )
FROM 'ri_index_selfRI.inp'
row delimited by '
'
LIMIT 14   SKIP 10
IGNORE CONSTRAINT UNIQUE 2, FOREIGN KEY 8
word skip 10 quotes off escapes off strip
off
```

• **Example 5** – Load data into table `t1` from the BCP character file `bcp_file.bcp` using the FORMAT BCP load option:

```sql
LOAD TABLE t1 (c1, c2, c3)
FROM 'bcp_file.bcp'
FORMAT BCP
```

• **Example 6** – Load default values 12345 into `c1` using the DEFAULT load option, and load `c2` and `c3` with data from the `LoadConst04.dat` file:

```sql
LOAD TABLE t1 (c1 DEFAULT '12345 ', c2, c3, filler(1))
FROM 'LoadConst04.dat'
STRIP OFF
QUOTES OFF
ESCAPES OFF
DELIMITED BY ',';
```

• **Example 7** – Load `c1` and `c2` with data from the file `bcp_file.bcp` using the FORMAT BCP load option and set `c3` to the value 10:

```sql
LOAD TABLE t1 (c1, c2, c3 DEFAULT '10')
FROM 'bcp_file.bcp'
FORMAT BCP
```
• **Example 8** – This code fragment ignores one header row at the beginning of the data file, where the header row is delimited by ‘&&’:

```
LOAD TABLE
...HEADER SKIP 1 HEADER DELIMITED by '&&'
```

• **Example 9** – This code fragment ignores 2 header rows at the beginning of the data file, where each header row is delimited by ‘\n’:

```
LOAD TABLE
...HEADER SKIP 2
```

• **Example 10** – Load a file into a RLV-enabled table.

```
Load data into RLV-enabled table rvt1 from the BCP character file bcp_file.bcp using the FORMAT BCP load option:
```

```
LOAD TABLE rvt1 (c1, c2, c3)
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

**Usage**

The `LOAD TABLE` statement allows efficient mass insertion into a database table from a file with ASCII or binary data.

The `LOAD TABLE` options also let you control load behavior when integrity constraints are violated and to log information about the violations.

You can use `LOAD TABLE` on a temporary table, but the temporary table must have been declared with `ON COMMIT PRESERVE ROWS`, or the next `COMMIT` removes the rows you have loaded.

You can also specify more than one file to load data. In the `FROM` clause, specify each `filename-string` separated by commas. Because of resource constraints, SAP Sybase IQ does not guarantee that all the data can be loaded. If resource allocation fails, the entire load transaction is rolled back. The files are read one at a time, and processed in the order specified in the `FROM` clause. Any `SKIP` or `LIMIT` value only applies in the beginning of the load, not for each file.

**Note:** When loading a multiplex database, use absolute (fully qualified) paths in all file names. Do not use relative path names.

`LOAD TABLE` supports loading of large object (LOB) data.

SAP Sybase IQ supports loading from both ASCII and binary data, and it supports both fixed- and variable-length formats. To handle all of these formats, you must supply a `load-specification` to tell SAP Sybase IQ what kind of data to expect from each “column” or field in the source file. The `column-spec` lets you define these formats:

**Advanced Security Options in SAP Sybase IQ**
• ASCII with a fixed length of bytes. The *input-width* value is an integer indicating the fixed width in bytes of the input field in every record.
• Binary or non-binary fields that use a number of PREFIX bytes (1, 2, or 4) to specify the length of the input.

There are two parts related to a **PREFIX** clause:
• Prefix value – always a binary value.
• Associated data bytes – always character format; never binary format.

If the data is unloaded using the extraction facility with the `TEMP_EXTRACT_BINARY` option set ON, you must use the **BINARY WITH NULL BYTE** parameter for each column when you load the binary data.

• Variable-length characters delimited by a separator. You can specify the terminator as hexadecimal ASCII characters. The **delimiter-string** can be any string of up to 4 characters, including any combination of printable characters, and any 8-bit hexadecimal ASCII code that represents a nonprinting character. For example, specify:
  • `\x09` to represent a tab as the terminator.
  • `\x00` for a null terminator (no visible terminator as in “C” strings).
  • `\x0a` for a newline character as the terminator. You can also use the special character combination of `\n` for newline.

**Note:** The delimiter string can be from 1 to 4 characters long, but you can specify only a single character in the **DELIMITED BY** clause. For **BCP**, the delimiter can be up to 10 characters.

• **DATE** or **DATETIME** string as ASCII characters. You must define the **input-date-format** or **input-datetime-format** of the string using one of the corresponding formats for the date and datetime data types supported by SAP Sybase IQ. Use **DATE** for date values and **DATETIME** for datetime and time values.

### Table 8. Formatting Dates and Times

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>yyyy or YYYY</td>
<td>Represents number of year. Default is current year.</td>
</tr>
<tr>
<td>yy or YY</td>
<td></td>
</tr>
<tr>
<td>mm or MM</td>
<td>Represents number of month. Always use leading zero or blank for number of the month where appropriate, for example, '05' for May. <strong>DATE</strong> value must include a month. For example, if the <strong>DATE</strong> value you enter is 1998, you receive an error. If you enter '03', SAP Sybase IQ applies the default year and day and converts it to '1998-03-01'.</td>
</tr>
<tr>
<td>dd or DD</td>
<td></td>
</tr>
<tr>
<td>jjj or JJJ</td>
<td>Represents number of day. Default day is 01. Always use leading zeros for number of day where appropriate, for example, '01' for first day. J or j indicates a Julian day (1 to 366) of the year.</td>
</tr>
</tbody>
</table>
Option | Meaning
--- | ---
hh | Represents hour. Hour is based on 24-hour clock. Always use leading zeros or blanks for hour where appropriate, for example, ‘01’ for 1 am. ‘00’ is also valid value for hour of 12 a.m.
HH | Represents hour. Hour is based on 24-hour clock. Always use leading zeros or blanks for hour where appropriate, for example, ‘01’ for 1 am. ‘00’ is also valid value for hour of 12 a.m.
nn | Represents minute. Always use leading zeros for minute where appropriate, for example, ‘08’ for 8 minutes.
ss[.ssssss] | Represents seconds and fraction of a second.
aa | Represents the a.m. or p.m. designation.
pp | Represents the p.m. designation only if needed. (This is an incompatibility with SAP Sybase IQ versions earlier than 12.0; previously, “pp” was synonymous with “aa”.)

SAP Sybase IQ assumes zero for minutes and seconds. For example, if the DATETIME value you enter is ‘03’, SAP Sybase IQ converts it to ‘03:00:00.0000’.

SAP Sybase IQ assumes zero for seconds. For example, if the time value you enter is ‘03:25’, SAP Sybase IQ converts it to ‘03:25:00.0000’.

Table 9. Sample DATE and DATETIME Format Options

<table>
<thead>
<tr>
<th>Input data</th>
<th>Format specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/31/98</td>
<td>DATE ('MM/DD/YY')</td>
</tr>
<tr>
<td>19981231</td>
<td>DATE ('YYYYMMDD')</td>
</tr>
<tr>
<td>123198140150</td>
<td>DATETIME ('MMDDYYhhmmss')</td>
</tr>
<tr>
<td>14:01:50 12-31-98</td>
<td>DATETIME ('hh:mm:ss MM-DD-YY')</td>
</tr>
<tr>
<td>18:27:53</td>
<td>DATETIME ('hh:mm:ss')</td>
</tr>
<tr>
<td>12/31/98 02:01:50AM</td>
<td>DATETIME ('MM/DD/YY hh:mm:ssaa')</td>
</tr>
</tbody>
</table>

SAP Sybase IQ has built-in load optimizations for common date, time, and datetime formats. If your data to be loaded matches one of these formats, you can significantly decrease load time by using the appropriate format.

You can also specify the date/time field as an ASCII fixed-width field (as described above) and use the FILLER(1) option to skip the column delimiter.

The NULL portion of the column-spec indicates how to treat certain input values as NULL values when loading into the table column. These characters can include BLANKS, ZEROS, or any other list of literals you define. When specifying a NULL value or reading a NULL value from the source file, the destination column must be able to contain NULLs.

ZEROS are interpreted as follows: the cell is set to NULL if (and only if) the input data (before conversion, if ASCII) is all binary zeros (and not character zeros).
• If the input data is character zero, then:
  1. NULL (ZEROS) never causes the cell to be NULL.
  2. NULL ('0') causes the cell to be NULL.
• If the input data is binary zero (all bits clear), then:
  1. NULL (ZEROS) causes the cell to be NULL.
  2. NULL ('0') never causes the cell to be NULL.

For example, if your LOAD statement includes `col1 date('yymmdd') null(zeros)` and the date is 000000, you receive an error indicating that 000000 cannot be converted to a DATE(4). To get LOAD TABLE to insert a NULL value in `col1` when the data is 000000, either write the NULL clause as `null('000000')`, or modify the data to equal binary zeros and use NULL(ZEROS).

If the length of a VARCHAR cell is zero and the cell is not NULL, you get a zero-length cell. For all other data types, if the length of the cell is zero, SAP Sybase IQ inserts a NULL. This is ANSI behavior. For non-ANSI treatment of zero-length character data, set the NON_ANSI_NULL_VARCHAR database option.

Use the DEFAULT option to specify a load default column value. You can load a default value into a column, even if the column does not have a default value defined in the table schema. This feature provides more flexibility at load time.

• The LOAD TABLE DEFAULTS option must be ON in order to use the default value specified in the LOAD TABLE statement. If the DEFAULTS option is OFF, the specified load default value is not used and a NULL value is inserted into the column instead.
• The LOAD TABLE command must contain at least one column that needs to be loaded from the file specified in the LOAD TABLE command. Otherwise, an error is reported and the load is not performed.
• The specified load default value must conform to the supported default values for columns and default value restrictions. The LOAD TABLE DEFAULT option does not support AUTOINCREMENT, IDENTITY, or GLOBAL AUTOINCREMENT as a load default value.
• The LOAD TABLE DEFAULT default-value must be of the same character set as that of the database.
• Encryption of the default value is not supported for the load default values specified in the LOAD TABLE DEFAULT clause.
• A constraint violation caused by evaluation of the specified load default value is counted for each row that is inserted in the table.

Another important part of the load-specification is the FILLER option. This option indicates you want to skip over a specified field in the source input file. For example, there may be characters at the end of rows or even entire fields in the input files that you do not want to add to the table. As with the column-spec definition, FILLER specifies ASCII fixed length of bytes, variable length characters delimited by a separator, and binary fields using PREFIX bytes.

The filename-string is passed to the server as a string. The string is therefore subject to the same formatting requirements as other SQL strings. In particular:
To indicate directory paths in Windows systems, the backslash character \ must be represented by two backslashes. Therefore, the statement to load data from the file c:\temp\input.dat into the Employees table is:

```
LOAD TABLE Employees
FROM 'c:\\temp\\input.dat' ...
```

The path name is relative to the database server, not to the client application. If you are running the statement on a database server on some other computer, the directory names refers to directories on the server machine, not on the client machine.

Descriptions of each statement clause follow:

**USING**— **USING FILE** loads one or more files from the server. This clause is synonymous with specifying the **FROM filename** clause. **USING CLIENT FILE** bulk loads one or more files from a client. The character set of the file on the client side must be the same as the server collation. SAP Sybase IQ serially processes files in the file list. Each file is locked in read mode as it is processed, then unlocked. Client-side bulk loading incurs no administrative overhead, such as extra disk space, memory or network-monitoring daemon requirements.

When bulk loading large objects, the **USING CLIENT FILE** clause applies to both primary and secondary files.

During client-side loads, the **IGNORE CONSTRAINT** log files are created on the client host and any error while creating the log files causes the operation to roll back.

Client-side bulk loading is supported by Interactive SQL and ODBC/JDBC clients using the Command Sequence protocol. It is not supported by clients using the TDS protocol. For data security over a network, use Transport Layer Security. To control who can use client-side bulk loads, use the secure feature (-sf) server startup switch, the **ALLOW_READ_CLIENT_FILE** database option, and/or the **READCLIENTFILE** access control.

The **LOAD TABLE FROM** clause is deprecated, but may be used to specify a file that exists on the server.

This example loads data from the file a.inp on a client computer.

```
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT_FILE 'c:\\client-data\\a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:\\client-data\\m.log'
ROW LOG 'c:\\client-data\\r.log'
ONLY LOG UNIQUE
```

**CHECK CONSTRAINTS**—This option defaults to ON. When you specify **CHECK CONSTRAINTS ON**, check constraints are evaluated and you are free to ignore or log them.

Setting **CHECK CONSTRAINTS OFF** causes SAP Sybase IQ to ignore all check constraint violations. This can be useful, for example, during database rebuilding. If a table has check constraints that call user-defined functions that are not yet created, the rebuild fails unless this option is set to OFF.
This option is mutually exclusive to the following options. If any of these options are specified in the same load, an error results:

- **IGNORE CONSTRAINT ALL**
- **IGNORE CONSTRAINT CHECK**
- **LOG ALL**
- **LOG CHECK**

**DEFAULTS**—If the **DEFAULTS** option is ON (the default) and the column has a default value, that value is used. If the **DEFAULTS** option is OFF, any column not present in the column list is assigned NULL.

The setting for the **DEFAULTS** option applies to all column DEFAULT values, including **AUTOINCREMENT**.

**QUOTES**—This parameter is optional and the default is ON. With **QUOTES** turned on, **LOAD TABLE** expects input strings to be enclosed in quote characters. The quote character is either an apostrophe (single quote) or a quotation mark (double quote). The first such character encountered in a string is treated as the quote character for the string. String data must be terminated with a matching quote.

With **QUOTES ON**, column or row delimiter characters can be included in the column value. Leading and ending quote characters are assumed not to be part of the value and are excluded from the loaded data value.

To include a quote character in a value with **QUOTES ON**, use two quotes. For example, this line includes a value in the third column that is a single quote character:

```
'123 High Street, Anytown', '(715)398-2354', ''''
```

With **STRIP** turned on (the default), trailing blanks are stripped from values before they are inserted. Trailing blanks are stripped only for non-quoted strings. Quoted strings retain their trailing blanks. Leading blank or TAB characters are trimmed only when the **QUOTES** setting is ON.

The data extraction facility provides options for handling quotes (**TEMP_EXTRACT_QUOTES**, **TEMP_EXTRACT_QUOTES_ALL**, and **TEMP_EXTRACT_QUOTE**). If you plan to extract data to be loaded into an IQ main store table and the string fields contain column or row delimiter under default ASCII extraction, use the **TEMP_EXTRACT_BINARY** option for the extract and the **FORMAT binary** and **QUOTES OFF** options for **LOAD TABLE**.

Limits:

- **QUOTES ON** applies only to column-delimited ASCII fields.
- With **QUOTES ON**, the first character of a column delimiter or row terminator cannot be a single or double quote mark.
- The **QUOTES** option does not apply to loading binary large object (**BLOB**) or character large object (**CLOB**) data from the secondary file, regardless of its setting. A leading or
trailing quote is loaded as part of CLOB data. Two consecutive quotes between enclosing quotes are loaded as two consecutive quotes with the QUOTES ON option.

- Adaptive Server Enterprise BCP does not support the QUOTES option. All field data is copied in or out equivalent to the QUOTES OFF setting. As QUOTES ON is the default setting for the SAP Sybase IQ LOAD TABLE statement, you must specify QUOTES OFF when importing ASE data from BCP output to an SAP Sybase IQ table.

Exceptions:

- If LOAD TABLE encounters any nonwhite characters after the ending quote character for an enclosed field, this error is reported and the load operation is rolled back:

  Non-SPACE text found after ending quote character for an enclosed field.
  SQLSTATE: QTA14  SQLCODE: -1005014L

- With QUOTES ON, if a single or double quote is specified as the first character of the column delimiter, an error is reported and the load operation fails:

  Single or double quote mark cannot be the 1st character of column delimiter or row terminator with QUOTES option ON.
  SQLSTATE: QCA90  SQLCODE: -1013090L

ESCAPES—If you omit a column-spec definition for an input field and ESCAPES is ON (the default), characters following the backslash character are recognized and interpreted as special characters by the database server. You can include newline characters as the combination \n, and other characters as hexadecimal ASCII codes, such as \x09 for the tab character. A sequence of two backslash characters (\) is interpreted as a single backslash. For SAP Sybase IQ, you must set ESCAPES OFF.

FORMAT—SAP Sybase IQ supports ASCII and binary input fields. The format is usually defined by the column-spec described above. If you omit that definition for a column, by default SAP Sybase IQ uses the format defined by this option. Input lines are assumed to have ascii (the default) or binary fields, one row per line, with values separated by the column delimiter character.

SAP Sybase IQ also accepts data from BCP character files as input to the LOAD TABLE command.

- The BCP data file loaded into SAP Sybase IQ tables using the LOAD TABLE FORMAT BCP statement must be exported (BCP OUT) in cross-platform file format using the -c option.
- For FORMAT BCP, the default column delimiter for the LOAD TABLE statement is <tab> and the default row terminator is <newline>.
- For FORMAT BCP, the last column in a row must be terminated by the row terminator, not by the column delimiter. If the column delimiter is present before the row terminator, then the column delimiter is treated as a part of the data.
- Data for columns that are not the last column in the load specification must be delimited by the column delimiter only. If a row terminator is encountered before a column delimiter for
a column that is not the last column, then the row terminator is treated as a part of the column data.

- Column delimiter can be specified via the `DELETED BY` clause. For `FORMAT BCP`, the delimiter must be less than or equal to 10 characters in length. An error is returned, if the delimiter length is more than 10.

- For `FORMAT BCP`, the load specification may contain only column names, `NULL`, and `ENCRYPTED`. An error is returned, if any other option is specified in the load specification.

For example, these `LOAD TABLE` load specifications are valid:

```sql
LOAD TABLE x( c1, c2 null(blanks), c3 )
FROM 'bcp_file.bcp'
FORMAT BCP
...

LOAD TABLE x( c1 encrypted(bigint,'KEY-ONE','aes'), c2, c3 )
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

`DELETED BY`—If you omit a column delimiter in the `column-spec` definition, the default column delimiter character is a comma. You can specify an alternative column delimiter by providing a single ASCII character or the hexadecimal character representation. The `DELETED BY` clause is:

```sql
... DELETED BY '\x09' ...
```

To use the newline character as a delimiter, you can specify either the special combination `\n` or its ASCII value `\x0a`. Although you can specify up to four characters in the column-spec `delimiter-string`, you can specify only a single character in the `DELETED BY` clause.

`STRIP`—The `STRIP` clause specifies whether unquoted values should have trailing blanks stripped off before they are inserted. The `LOAD TABLE` command accepts these `STRIP` keywords:

- `STRIP OFF`—Do not strip off trailing blanks.
- `STRIP RTRIM`—Strip trailing blanks.
- `STRIP ON`—Deprecated. Use `STRIP RTRIM`.

With `STRIP` turned on (the default), SAP Sybase IQ strips trailing blanks from values before inserting them. This is effective only for `VARCHAR` data. `STRIP OFF` preserves trailing blanks.

Trailing blanks are stripped only for unquoted strings. Quoted strings retain their trailing blanks. If you do not require blank sensitivity, you can use the `FILLER` option as an alternative to be more specific in the number of bytes to strip, instead of all the trailing spaces. `STRIP OFF` is more efficient for SAP Sybase IQ, and it adheres to the ANSI standard when dealing with trailing blanks. (CHAR data is always padded, so the `STRIP` option only affects VARCHAR data.)

The `STRIP` option applies only to variable-length non-binary data and does not apply to ASCII fixed-width inserts. For example, assume this schema:
CREATE TABLE t( c1 VARCHAR(3) );
LOAD TABLE t( c1 ',,' ) .......... STRIP RTRIM     // trailing blanks trimmed
LOAD TABLE t( c1 ',,' ) .......... STRIP OFF      // trailing blanks not trimmed
LOAD TABLE t( c1 ASCII(3) ) ... STRIP RTRIM     // trailing blanks not trimmed
LOAD TABLE t( c1 ASCII(3) ) ... STRIP OFF       // trailing blanks trimmed
LOAD TABLE t( c1 BINARY ) ...... STRIP RTRIM     // trailing blanks trimmed
LOAD TABLE t( c1 BINARY ) ...... STRIP OFF       // trailing blanks trimmed

Trailing blanks are always trimmed from binary data.

WITH CHECKPOINT—This option is useful only when loading SQL Anywhere tables in an SAP Sybase IQ database.

Use this clause to specify whether to perform a checkpoint. The default setting is OFF. If this clause is set to ON, a checkpoint is issued after successfully completing and logging the statement. If the server fails after a connection commits and before the next checkpoint, the data file used to load the table must be present for the recovery to complete successfully. However, if WITH CHECKPOINT ON is specified, and recovery is subsequently required, the data file need not be present at the time of recovery.

The data files are required, regardless of what is specified for this clause, if the database becomes corrupt and you need to use a backup and apply the current log file.

Warning! If you set the database option CONVERSION_ERROR to OFF, you may load bad data into your table without any error being reported. If you do not specify WITH CHECKPOINT ON, and the database needs to be recovered, the recovery may fail as CONVERSION_ERROR is ON (the default value) during recovery. It is recommended that you do not load tables when CONVERSION_ERROR is set to OFF and WITH CHECKPOINT ON is not specified.

See also CONVERSION_ERROR Option [TSQL].

BYTE ORDER—Specifies the byte order during reads. This option applies to all binary input fields. If none are defined, this option is ignored. SAP Sybase IQ always reads binary data in the format native to the machine it is running on (default is NATIVE). You can also specify:

- **HIGH** when multibyte quantities have the high order byte first (for big endian platforms like Sun, IBM AIX, and HP).
- **LOW** when multibyte quantities have the low order byte first (for little endian platforms like Windows).

LIMIT—Specifies the maximum number of rows to insert into the table. The default is 0 for no limit. The maximum is $2^{31} - 1$ (2147483647) rows.
NOTIFY—Specifies that you be notified with a message each time the specified number of rows is successfully inserted into the table. The default is every 100,000 rows. The value of this option overrides the value of the NOTIFY_MODULUS database option.

ON FILE ERROR—Specifies the action SAP Sybase IQ takes when an input file cannot be opened because it does not exist or you have incorrect permissions to read the file. You can specify one of the following:

- **ROLLBACK** aborts the entire transaction (the default).
- **FINISH** finishes the insertions already completed and ends the load operation.
- **CONTINUE** returns an error but only skips the file to continue the load operation.

Only one ON FILE ERROR clause is permitted.

PREVIEW—Displays the layout of input into the destination table including starting position, name, and data type of each column. SAP Sybase IQ displays this information at the start of the load process. If you are writing to a log file, this information is also included in the log.

ROW DELIMITED BY—Specifies a string up to 4 bytes in length that indicates the end of an input record. You can use this option only if all fields within the row are any of the following:

- Delimited with column terminators
- Data defined by the DATE or DATETIME column-spec options
- ASCII fixed length fields

You cannot use this option if any input fields contain binary data. With this option, a row terminator causes any missing fields to be set to NULL. All rows must have the same row delimiters, and it must be distinct from all column delimiters. The row and field delimiter strings cannot be an initial subset of each other. For example, you cannot specify “*#” as a field delimiter and “#*” as the row delimiter, but you could specify “*#” as the field delimiter with that row delimiter.

If a row is missing its delimiters, SAP Sybase IQ returns an error and rolls back the entire load transaction. The only exception is the final record of a file where it rolls back that row and returns a warning message. On Windows, a row delimiter is usually indicated by the newline character followed by the carriage return character. You might need to specify this as the delimiter-string (see above for description) for either this option or FILLER.

SKIP—Defines the number of rows to skip at the beginning of the input tables for this load. The maximum number of rows to skip is $2^{31} - 1$ (2147483647). The default is 0.

HEADER SKIP…HEADER DELIMITED BY—Specifies a number of lines at the beginning of the data file, including header rows, for LOAD TABLE to skip. All LOAD TABLE column specifications and other load options are ignored, until the specified number of rows is skipped.

- The number of lines to skip is greater than or equal to zero.
• Lines are determined by a 1 to 4 character delimiter string specified in the HEADER DELIMITED BY clause. The default HEADER DELIMITED BY string is the ‘\n’ character.
• The HEADER DELIMITED BY string has a maximum length of four characters. An error is returned, if the string length is greater than four or less than one.
• When a non-zero HEADER SKIP value is specified, all data inclusive of the HEADER DELIMITED BY delimiter is ignored, until the delimiter is encountered the number of times specified in the HEADER SKIP clause.
• All LOAD TABLE column specifications and other load options are ignored, until the specified number of rows has been skipped. After the specified number of rows has been skipped, the LOAD TABLE column specifications and other load options are applied to the remaining data.
• The "header" bytes are ignored only at the beginning of the data. When multiple files are specified in the USING clause, HEADER SKIP only ignores data starting from the first row of the first file, until it skips the specified number of header rows, even if those rows exist in subsequent files. LOAD TABLE does not look for headers once it starts parsing actual data.
• No error is reported, if LOAD TABLE processes all input data before skipping the number of rows specified by HEADER SKIP.

WORD SKIP—Allows the load to continue when it encounters data longer than the limit specified when the word index was created.

If a row is not loaded because a word exceeds the maximum permitted size, a warning is written to the .iqmsg file. WORD size violations can be optionally logged to the MESSAGE LOG file and rejected rows logged to the ROW LOG file specified in the LOAD TABLE statement.

• If the option is not specified, LOAD TABLE reports an error and rolls back on the first occurrence of a word that is longer than the specified limit.
• number specifies the number of times the “Words exceeding the maximum permitted word length not supported” error is ignored.
• 0 (zero) means there is no limit.

ON PARTIAL INPUT ROW—Specifies the action to take when a partial input row is encountered during a load. You can specify one of the following:

• CONTINUE issues a warning and continues the load operation. This is the default.
• ROLLBACK aborts the entire load operation and reports the error.

Partial input record skipped at EOF.
SQLSTATE: QDC32   SQLSTATE: -1000232L

IGNORE CONSTRAINT—Specifies whether to ignore CHECK, UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations that occur during a load and the maximum number of violations to ignore before initiating a rollback. Specifying each constraint type has the following result:

• CHECK limit—If limit specifies zero, the number of CHECK constraint violations to ignore is infinite. If CHECK is not specified, the first occurrence of any CHECK constraint
violation causes the **LOAD** statement to roll back. If *limit* is nonzero, then the *limit* + 1 occurrence of a CHECK constraint violation causes the load to roll back.

- **UNIQUE limit**—If *limit* specifies zero, then the number of UNIQUE constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit* + 1 occurrence of a UNIQUE constraint violation causes the load to roll back.

- **NULL limit**—If *limit* specifies zero, then the number of NULL constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit* + 1 occurrence of a NULL constraint violation causes the load to roll back.

- **FOREIGN KEY limit**—If *limit* specifies zero, the number of FOREIGN KEY constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit* + 1 occurrence of a FOREIGN KEY constraint violation causes the load to roll back.

- **DATA VALUE limit**—If the database option `CONVERSION_ERROR = ON`, an error is reported and the statement rolls back. If *limit* specifies zero, then the number of DATA VALUE constraint violations (data type conversion errors) to ignore is infinite. If *limit* is nonzero, then the *limit* + 1 occurrence of a DATA VALUE constraint violation causes the load to roll back.

- **ALL limit**—If the database option `CONVERSION_ERROR = ON`, an error is reported and the statement rolls back. If *limit* specifies zero, then the cumulative total of all integrity constraint violations to ignore is infinite. If *limit* is nonzero, then load rolls back when the cumulative total of all ignored UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations exceeds the value of *limit*. For example, you specify this **IGNORE CONSTRAINT** option:

```
IGNORE CONSTRAINT NULL 50, UNIQUE 100, ALL 200
```

The total number of integrity constraint violations cannot exceed 200, whereas the total number of NULL and UNIQUE constraint violations cannot exceed 50 and 100, respectively. Whenever any of these limits is exceeded, the **LOAD TABLE** statement rolls back.

**Note:** A single row can have more than one integrity constraint violation. Every occurrence of an integrity constraint violation counts towards the limit of that type of violation.

**Tip:** Set the **IGNORE CONSTRAINT** option limit to a nonzero value if you are logging the ignored integrity constraint violations. Logging an excessive number of violations affects the performance of the load.

If CHECK, UNIQUE, NULL, or FOREIGN KEY is not specified in the **IGNORE CONSTRAINT** clause, then the load rolls back on the first occurrence of each of these types of integrity constraint violation.

If DATA VALUE is not specified in the **IGNORE CONSTRAINT** clause, then the load rolls back on the first occurrence of this type of integrity constraint violation, unless the database option `CONVERSION_ERROR = OFF`. If `CONVERSION_ERROR = OFF`, a warning is reported for any DATA VALUE constraint violation and the load continues.
When the load completes, an informational message regarding integrity constraint violations is logged in the .iqmsg file. This message contains the number of integrity constraint violations that occurred during the load and the number of rows that were skipped.

MESSAGE LOG—Specifies the names of files in which to log information about integrity constraint violations and the types of violations to log. Timestamps indicating the start and completion of the load are logged in both the MESSAGE LOG and the ROW LOG files. Both MESSAGE LOG and ROW LOG must be specified, or no information about integrity violations is logged.

- If the ONLY LOG clause is not specified, no information on integrity constraint violations is logged. Only the timestamps indicating the start and completion of the load are logged.
- Information is logged on all integrity constraint-type violations specified in the ONLY LOG clause or for all word index-length violations if the keyword WORD is specified.
- If constraint violations are being logged, every occurrence of an integrity constraint violation generates exactly one row of information in the MESSAGE LOG file.

The number of rows (errors reported) in the MESSAGE LOG file can exceed the IGNORE CONSTRAINT option limit, because the load is performed by multiple threads running in parallel. More than one thread might report that the number of constraint violations has exceeded the specified limit.

- If constraint violations are being logged, exactly one row of information is logged in the ROW LOG file for a given row, regardless of the number of integrity constraint violations that occur on that row.

The number of distinct errors in the MESSAGE LOG file might not exactly match the number of rows in the ROW LOG file. The difference in the number of rows is due to the parallel processing of the load described above for the MESSAGE LOG.

- The MESSAGE LOG and ROW LOG files cannot be raw partitions or named pipes.

- If the MESSAGE LOG or ROW LOG file already exists, new information is appended to the file.

- Specifying an invalid file name for the MESSAGE LOG or ROW LOG file generates an error.

- Specifying the same file name for the MESSAGE LOG and ROW LOG files generates an error.

Various combinations of the IGNORE CONSTRAINT and MESSAGE LOG options result in different logging actions.
Table 10. LOAD TABLE Logging Actions

<table>
<thead>
<tr>
<th>IGNORE CONSTRAINT specified?</th>
<th>MESSAGE LOG specified?</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>yes</td>
<td>All ignored integrity constraint violations are logged, including the user specified limit, before the rollback.</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>The first integrity constraint violation is logged before the rollback.</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>Nothing is logged.</td>
</tr>
<tr>
<td>no</td>
<td>no</td>
<td>Nothing is logged. The first integrity constraint violation causes a rollback.</td>
</tr>
</tbody>
</table>

**Tip:** Set the **IGNORE CONSTRAINT** option limit to a nonzero value, if you are logging the ignored integrity constraint violations. If a single row has more than one integrity constraint violation, a row for each violation is written to the **MESSAGE LOG** file. Logging an excessive number of violations affects the performance of the load.

**LOG DELIMITED BY**—Specifies the separator between data values in the **ROW LOG** file. The default separator is a comma.

SAP Sybase IQ no longer returns an error message when **FORMAT BCP** is specified as a **LOAD TABLE** clause. In addition, these conditions are verified and proper error messages are returned:

- If the specified load format is not **ASCII, BINARY, or BCP**, SAP Sybase IQ returns the message “Only ASCII, BCP and BINARY are supported LOAD formats.”
- If the **LOAD TABLE** column specification contains anything other than column name, **NULL, or ENCRYPTED**, then SAP Sybase IQ returns the error message “Invalid load specification for LOAD ... FORMAT BCP.”
- If the column delimiter or row terminator size for the **FORMAT BCP** load is greater than 10 characters, then SAP Sybase IQ returns the message “Delimiter '%2' must be 1 to %3 characters in length.” (where %3 equals 10).

Messages corresponding to error or warning conditions which can occur for **FORMAT BCP** as well as **FORMAT ASCII** are the same for both formats.

- If the load default value specified is **AUTOINCREMENT, IDENTITY, or GLOBAL AUTOINCREMENT**, SAP Sybase IQ returns the error “Default value %2 cannot be used as a LOAD default value. %1”
- If the **LOAD TABLE** specification does not contain any columns that need to be loaded from the file specified, SAP Sybase IQ returns the error “The LOAD statement must
contain at least one column to be loaded from input file.” and the LOAD TABLE statement rolls back.

- If a load exceeds the limit on the maximum number of terms for a text document with TEXT indexes, SAP Sybase IQ returns the error “Text document exceeds maximum number of terms. Support up to 4294967295 terms per document.”

**Standards**

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

**Permissions**

The permissions required to execute a LOAD TABLE statement depend on the database server -gl command line option, as follows:

- -gl ALL – You must be the owner of the table, have ALTER or LOAD permission on the table, or have the ALTER ANY TABLE, LOAD ANY TALBE, or ALTER ANY OBJECT system privilege.
- -gl DBA – You must have the ALTER ANY TABLE, LOAD ANY TABLE, or ALTER ANY OBJECT system privilege.
- -gl NONE – Execution of the LOAD TABLE statement is not permitted.

For more information on the -gl command line option, please refer Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options.

LOAD TABLE also requires a write lock on the table.

**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 Sybase 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 Sybase 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.

Database Options for Column Encryption

Certain SAP Sybase IQ database option settings affect column encryption and decryption; the default settings are not optimal for most column encryption operations.

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.

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.
**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:

```sql
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 Sybase IQ earlier than 12.7.

**CONVERSION_MODE Option**

Restricts implicit conversion between binary data types (BINARY, VARBINARY, and LONG BINARY) and other non-binary data types (BIT, TINYINT, SMALLINT, INT, UNSIGNED INT, BIGINT, UNSIGNED BIGINT, CHAR, VARCHAR, and LONG VARCHAR) on various operations.

**Allowed Values**

0, 1

**Default**

0

**Scope**

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

**Description**

The default value of 0 maintains implicit conversion behavior prior to version 12.7. Setting `CONVERSION_MODE` to 1 restricts implicit conversion of binary data types to any other non-binary data type on `INSERT`, `UPDATE`, and in queries. The restrict binary conversion mode also applies to `LOAD TABLE` default values and `CHECK` constraint. The use of this option
prevents implicit data type conversions of encrypted data that would result in semantically meaningless operations.

Users must be specifically licensed to use the encrypted column functionality of the SAP Sybase IQ Advanced Security Option.

**Implicit Conversion Restrictions**
The `CONVERSION_MODE` option restrict binary mode value of 1 (`CONVERSION_MODE = 1`) restricts implicit conversion for these operations:

- **LOAD TABLE** with **CHECK** constraint or default value
- **INSERT...SELECT, INSERT...VALUE, and INSERT...LOCATION**
- Certain types of **UPDATE**
- Certain types of **INSERT** and **UPDATE** via updatable cursor
- All aspects of queries in general

**Encryption and Decryption Example**
An example using the `AES_ENCRYPT` and `AES_DECRYPT` functions, written in commented SQL.

```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 (HighGroup) 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):

```
<table>
<thead>
<tr>
<th>PlainText</th>
<th>CipherText</th>
<th>Corresponding Numeric Precisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>numeric(1,0) - numeric(20,0)</td>
</tr>
<tr>
<td>1 - 16</td>
<td>32</td>
<td>numeric(21,0) - numeric(52,0)</td>
</tr>
<tr>
<td>17 - 32</td>
<td>48</td>
<td>numeric(53,0) - numeric(84,0)</td>
</tr>
<tr>
<td>33 - 48</td>
<td>64</td>
<td>numeric(85,0) - numeric(116,0)</td>
</tr>
<tr>
<td>49 - 64</td>
<td>80</td>
<td>numeric(117,0) - numeric(128,0)</td>
</tr>
<tr>
<td>65 - 80</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>81 - 96</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>97 - 112</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>113 - 128</td>
<td>144</td>
<td></td>
</tr>
<tr>
<td>129 - 144</td>
<td>160</td>
<td></td>
</tr>
<tr>
<td>145 - 160</td>
<td>176</td>
<td></td>
</tr>
<tr>
<td>161 - 176</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>177 - 192</td>
<td>208</td>
<td></td>
</tr>
<tr>
<td>193 - 208</td>
<td>224</td>
<td></td>
</tr>
<tr>
<td>209 - 224</td>
<td>240</td>
<td></td>
</tr>
</tbody>
</table>
```

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
    from DBA.first_table
      cross join DBA.second_table
      and I.encryptedA = II.encryptedA
      and SUBSTR(I.sensitiveA,4,3)
      BETWEEN '345' AND '456';
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 '
')
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_function_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_decrypt_function_view set hidden;
grant select on DBA.second_decrypt_function_view to NonPrivUser;
grant select on DBA.second_decrypt_function_view to PrivUser;
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

**See also**
- *AES_ENCRYPT Function [String]* on page 195
- *AES_DECRYPT Function [String]* on page 198
- *LOAD TABLE ENCRYPTED Clause* on page 199

**Kerberos Authentication Support in SAP Sybase IQ**

SAP Sybase 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 Sybase IQ Advanced Security Option.

**Licensing Requirements for Kerberos**

The Advanced Security Option (IQ_SECURITY) protects your environment against unauthorized access, and is required to use Kerberos authentication with SAP Sybase IQ.

**LDAP User Authentication Support in SAP Sybase IQ**

You can integrate SAP Sybase IQ into any existing enterprise-wide directory access framework based on Lightweight Directory Access Protocol (LDAP), a widely accepted international standard.

**License Requirements for LDAP User Authentication**

The Advanced Security Option (IQ_SECURITY) protects your environment against unauthorized access, and is required to allow LDAP user authentication with SAP Sybase IQ.
Appendix: SQL Reference

Reference material for SQL statements, database options, functions, and system procedures mentioned in this document.

SQL Statements

Interactive SQL statements customize and modify the database.

ALTER LDAP SERVER Statement

Any changes to an LDAP server configuration object are applied on subsequent connections. Any connection already started when the change is applied does not immediately reflect the change.

In addition to resetting LDAP server configuration object values for attributes, the ALTER LDAP SERVER statement allows an administrator to make manual adjustments to a server's state and behavior by putting the LDAP server configuration object in maintenance mode and returning it to service from maintenance mode.

Syntax

```
ALTER LDAP SERVER <ldapua-server-name>
{ <ldapua-server-attribs>
  | [ WITH ( SUSPEND | ACTIVATE | REFRESH ) ] }
```

```
luxapua-server-attribs:
  SEARCH DN
    URL { 'URL_string' | NULL }
    | ACCESS ACCOUNT { 'DN_string' | NULL }
    | IDENTIFIED BY ( 'password>' | NULL }
    | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
  | AUTHENTICATION URL { 'URL_string' | NULL }
  | CONNECTION TIMEOUT timeout_value
  | CONNECTION RETRIES retry_value
  | TLS { ON | OFF }
```

Parameters

- **URL** – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP URL syntax before it is stored in the ISYSLDAPSERVER system table. The maximum size for this string is 1024 bytes.

- **ACCESS ACCOUNT** – a user created on the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect
to the LDAP server. This user has permissions within the LDAP server to search for DNs by user ID in the locations specified by the SEARCH DN URL. The maximum size for this string is 1024 bytes.

- **IDENTIFIED BY** – provides the password associated with the ACCESS ACCOUNT user. The password is stored using symmetric encryption on disk. Use the value NULL to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.

- **IDENTIFIED BY ENCRYPTED** – configures the password associated with the ACCESS ACCOUNT distinguished name in an encrypted format. The binary value is the encrypted password and is stored on disk as is. Use the value NULL to clear the password and set it to none. The maximum size of the binary is 289 bytes. The encrypted key should be a valid varbinary value. Do not enclose the encrypted key in quotation marks.

- **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for `<URL_string>` and is validated for correct LDAP URL syntax before it is stored in ISYSLDAPSERVER system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.

- **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.

- **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase 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 for both DN searches and authentication. When set to ON, the TLS protocol is used and the URL begins with "ldap://" When set to OFF (or not specified), Secure LDAP protocol is used and the URL begins with “ldaps://”. When using the TLS protocol, specify the database security option TRUSTED_CERTIFICATES_FILE with a file name containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.

- **WITH ACTIVATE** – sets the state of the LDAP server configuration object to READY and allows authentication with the LDAP server. Server option values are read from the ISYSLDAPSERVER system table and applied to new connections to the LDAP server and incoming authentication requests to the SAP Sybase IQ server. Upon successful authentication of a user, the state changes to ACTIVE.

- **WITH SUSPEND** – sets the state of the LDAP server configuration object to SUSPENDED, which puts the LDAP server configuration object in maintenance mode. Connections to the LDAP server are closed and authentication with the LDAP server becomes unavailable.

- **WITH REFRESH** – reinitializes LDAP user authentication. This command does not change the state of the LDAP server configuration object, nor does it change any existing connections from a client to the SAP Sybase IQ server. This parameter is typically used
with an LDAP server that is in an ACTIVE or READY state to release any resources that may be held or to reread changes made to files outside of the server, such as a change to the contents of the file specified by database option TRUSTED_CERTIFICATES_FILE.

**Note:** When the LDAP server is in any state other than ACTIVE or READY, REFRESH has no effect.

### Examples

- **Example 1** – suspends the LDAP server configuration object named `apps_primary`.

  ```sql
  ALTER LDAP SERVER apps_primary SUSPEND
  ```

- **Example 2** – changes the LDAP server configuration object named `apps_primary` to use a different URL for authentication on host `fairfax`, sets the port number to `1066`, sets the number of connection retries to `10`, and finally activates the LDAP server configuration object.

  ```sql
  ALTER LDAP SERVER apps_primary
  AUTHENTICATION URL 'ldap://my_LDAPserver:1066/'
  CONNECTION RETRIES 10
  WITH ACTIVATE
  ```

### Standards

ANSI SQL – Compliance level: Transact-SQL extension.

### Permissions

Requires the MANAGE ANY LDAP SERVER system privilege.

### ALTER LOGIN POLICY Statement

Changes existing login policies or configures logical server access.

### Syntax

**Syntax 1**

```sql
ALTER LOGIN POLICY policy-name
{ { ADD | DROP | SET } LOGICAL SERVER ls-assignment-list
[ LOGICAL SERVER ls-override-list ]
```

- **ls-assignment-list** –
  ```sql
  { { ls-name, ... } | ALL | COORDINATOR | SERVER | NONE | DEFAULT }
  ```

- **ls-override-list** –
  ```sql
  { ls-name, ... }
  ```

- **ls-name** –
  ```sql
  { OPEN | user-defined-ls-name }
  ```
• **policy-option-value**
  ```plaintext
  { UNLIMITED | DEFAULT | value }
  ```

Syntax 2

```
ALTER LOGIN POLICY  policy-name
AUTO_UNLOCK_TIME=0 – UNLIMITED
| CHANGE_PASSWORD_DUAL_CONTROL=[ON | OFF]
| DEFAULT_LOGICAL_SERVER=[logical_server_name | ALL | AUTO | COORDINATOR | NONE | OPEN | SERVER]
| LOCKED=[ON | OFF]
| MAX_CONNECTIONS=0 – 2147483647
| MAX_DAYS_SINCE_LOGIN=0 – 2147483647
| MAX_FAILED_LOGIN_ATTEMPTS=0 – 2147483647
| MAX_NON_DBA_CONNECTIONS=0 – 2147483647
| PASSWORD_EXPIRY_ON_NEXT_LOGIN=[ON | OFF]
| PASSWORD_GRACE_TIME=0 – 2147483647
| PASSWORD_LIFE_TIME=0 – 2147483647
| ROOT_AUTO_UNLOCK_TIME=0 – UNLIMITED
| LDAP_PRIMARY_SERVER=server_name
| LDAP_SECONDARY_SERVER=server_name
| LDAP_AUTO_FAILBACK_PERIOD=0 – 2147483647
| LDAP_FAIlOVER_TO_STD=[ON | OFF]
| LDAP_REFRESH_ON=NOW
```

**Applies to**

Simplex and multiplex.

**Examples**

• **Example 1** – see *Logical Server Access Configuration* and *Multiplex Login Policy Configuration*.

• **Example 2** – sets the password_life_time value to UNLIMITED and the max_failed_login_attempts value to 5 in the Test1 login policy.

```
ALTER LOGIN POLICY Test1
password_life_time=UNLIMITED
max_failed_login_attempts=5;
```

**Permissions**

Requires the MANAGE ANY LOGIN POLICY system privilege.
## Login Policy Options
Available options for root and user-defined login policies.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO_UNLOCK_TIME</td>
<td>The time period after which locked accounts not granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined in any login policy, including the root login policy.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – 0 – unlimited</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for Root policy</strong> – Unlimited</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users not granted the MANAGE ANY USER system privilege.</td>
</tr>
<tr>
<td>CHANGE_PASSWORD_DUAL_CONF</td>
<td>Requires input from two users, each granted the CHANGE PASSWORD system privilege, to change the password of another user.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – ON, OFF</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for Root policy</strong> – OFF</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users.</td>
</tr>
<tr>
<td>DEFAULT_LOGICAL_SERVER</td>
<td>If the connection string specifies no logical server, the user connects to the DEFAULT_LOGICAL_SERVER setting specified in the user's login policy.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> –</td>
</tr>
<tr>
<td></td>
<td>• Name of an existing user-defined logical server</td>
</tr>
<tr>
<td></td>
<td>• ALL – allows access to all logical servers.</td>
</tr>
<tr>
<td></td>
<td>• AUTO – value of the default logical server in the root login policy.</td>
</tr>
<tr>
<td></td>
<td>• COORDINATOR – the current coordinator node.</td>
</tr>
<tr>
<td></td>
<td>• NONE – denies access to any multiplex server.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• SERVER – allows access to all of the multiplex nodes, subject to the semantics of the SERVER logical server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for Root policy</strong> – AUTO</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users. Requires MANAGE MULTIPLEX system privilege.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LOCKED</td>
<td>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.</td>
</tr>
</tbody>
</table>
|                             | • **Values** – ON, OFF  
|                             | • **Initial value for Root policy** – 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  
|                             | • **Initial value for Root policy** – 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  
|                             | • **Initial value for Root policy** – 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 into the user account before the account is locked.          |
|                             | • **Values** – 0 – 2147483647  
|                             | • **Initial value for Root policy** – Unlimited  
|                             | • **Applies to** – All users.                                                                                                              |
| 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  
|                             | • **Initial value for Root policy** – Unlimited  
<p>|                             | • <strong>Applies to</strong> – All users except those with the SERVER OPERATOR or DROP CONNECTION privilege.                                      |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| PASSWORD_EXPIRY_ON_NEXT_LOGIN | If set ON, the user's password expires at the next login.  
  - **Values** – ON, OFF  
  - **Initial value for Root policy** – OFF  
  - **Applies to** – All users.  
  
  **Note:** This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL). |
| PASS-WORD_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  
  - **Initial value for Root policy** – 0  
  - **Applies to** – All users. |
| PASS-WORD_LIFE_TIME | The maximum number of days before a password must be changed.  
  - **Values** – 0 – 2147483647  
  - **Initial value for Root policy** – Unlimited  
  - **Applies to** – All users. |
| ROOT_AUTO_UNLOCK_TIME | The time period after which locked accounts granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined only in the root login policy.  
  - **Values** – 0 – unlimited  
  - **Initial value for Root policy** – 15  
  - **Applies to** – All users granted the MANAGE ANY USER system privilege. |

**LDAP Login Policy Options**  
Available login policy options for LDAP user authentication

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| LDAP_PRIMARY_SERVER | Specifies the name of the primary LDAP server.  
  - **Values** – n/a  
  - **Initial value for Root policy** – None  
  - **Applies to** – All users. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP_SECONDARY_SERVER</td>
<td>Specifies the name of the secondary LDAP server.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – n/a</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for ROOT policy</strong> – None</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users.</td>
</tr>
<tr>
<td>LDAP_AUTOFAILBACK_PERIOD</td>
<td>Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – 0 - 2147483647</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for ROOT policy</strong> – 15 minutes</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users.</td>
</tr>
<tr>
<td>LDAP_FAILOVER_TO_STD</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – ON, OFF</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for ROOT policy</strong> – ON</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users.</td>
</tr>
<tr>
<td>LDAP_REFRESH_DN</td>
<td>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 the option ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the user_dn value 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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Values</strong> – NOW</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for ROOT policy</strong> – NULL</td>
</tr>
<tr>
<td></td>
<td>• <strong>Initial value for user-defined login policy</strong> – Current time stored in UTC</td>
</tr>
<tr>
<td></td>
<td>• <strong>Applies to</strong> – All users.</td>
</tr>
</tbody>
</table>
Multiplex Login Policy Configuration
Configure login policies for multiplex servers.

Example
This example overrides the login policy settings on a logical server, increasing the maximum number of connections on logical server ls1:

```
ALTER LOGIN POLICY lp1 max_connections=20 LOGICAL SERVER ls1;
```

Usage
Applies only to multiplex.

Any login management commands you execute on any multiplex server automatically propagate to all servers in the multiplex. For best performance, execute these commands, or any DDL, on the coordinator.

An override at the logical server level override means that a particular login policy option has different settings for different logical servers. SYS.ISYSQLException stores login policy option values for logical-server override. For each logical-server override of a login policy option, a corresponding row exists in ISYSQLException.

Logical Server Access Configuration
Configure logical server access.

Example 1
Assume that the root login policy allows access to logical servers ls4 and ls5 and login policy lp1 exists with no logical server assignment. The statement below effectively assigns login policy lp1 to logical servers ls4 and ls5.

Assign logical server ls1 to login policy lp1:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls1
```

Example 2
This statement allows access of logical servers ls2 and ls3 from login policy lp1:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls2, ls3
```

Example 3
Modify login policy lp1 to allow access to ls3 and ls4 only:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls4
ALTER LOGIN POLICY lp1 DROP LOGICAL SERVER ls1, ls2
```

or:

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER ls3, ls4
```
Example 4
Modify login policy lp1 to deny access to any logical servers:
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER NONE

Example 5
Drop current logical server assignments of login policy lp1 and allow it to inherit the logical server assignments of the root login policy:
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER DEFAULT

Usage
ADD, DROP, or SET clauses let you configure the logical server assignments of a login policy:

- ADD – adds new logical server assignments to a login policy.
- DROP – deletes existing logical server assignments from a login policy.
- SET – replaces all logical server assignments for a login policy with a new set of logical server.

Use only one ADD, DROP, or SET clause. Use SERVER, NONE, and DEFAULT only with the SET clause. Specify a particular logical server name only once per ls-assignment list or ls-override list.

An error is returned if:

- Any logical server specified with the ADD clause is already assigned to the login policy.
- Any logical server specified with the DROP clause is currently not assigned to the login policy.
- Logical server assignment change may cause a membership overlap among assigned logical servers.

SYS.ISYSIQLOGINPOLICYLSINFO stores logical server assignment information. For each logical-server override of a login policy option, a corresponding row exists in ISYSIQLOGINPOLICYLSINFO.

ALTER ROLE Statement
Migrates a compatibility role to a user-defined system role, then automatically drops the compatibility role.

Note: You cannot use the ALTER ROLE statement to migrate SYS_AUTH_REMOTE_DBA_ROLE, SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE. SYS_AUTH_SA_ROLE and SYS_AUTH_SSO_ROLE are automatically migrated when SYS_AUTH_DBA_ROLE is migrated. SYS_AUTH_REMOTE_DBA_ROLE cannot be migrated at all.
Syntax

```sql
ALTER ROLE predefined_sys_role_name
MIGRATE TO new_role_name [, new_sa_role_name, new_sso_role_name]
```

Parameters

- **predefined_sys_role_name** – the name of a compatibility role that still exists (has not already been dropped) in the database.
- **new_role_name** – the name of the new role cannot begin with the prefix SYS_ or end with the suffix _ROLE.
- **new_sa_role_name** – required only when migrating SYS_AUTH_DBA_ROLE. The new role to which the underlying system privileges of SYS_AUTH_SA_ROLE are to be migrated to cannot already exist in the database, and the new role name cannot begin with the prefix SYS_ or end with the suffix _ROLE.
- **new_sso_role_name** – required only when migrating SYS_AUTH_DBA_ROLE. The new role to which the underlying system privileges of SYS_AUTH_SSO_ROLE are to be migrated to cannot already exist in the database, and the new role name cannot begin with the prefix SYS_ or end with the suffix _ROLE.

Examples

- **Example 1** – this statement migrates SYS_AUTH_DBA_ROLE to the new roles Custom_DBA, Custom_SA, and Custom_SSO respectively. It then automatically migrates all users, underlying system privileges, and roles granted to SYS_AUTH_DBA_ROLE to the applicable new roles. Finally, it drops SYS_AUTH_DBA_ROLE, SYS_AUTH_SA_ROLE, and SYS_AUTH_SSO_ROLE.

```sql
ALTER ROLE SYS_AUTH_DBA_ROLE
MIGRATE TO Custom_DBA, Custom_SA, Custom_SSO
```

- **Example 2** – this statement migrates SYS_AUTH_OPERATOR_ROLE role to the new role Operator_role. It then automatically migrates all users, underlying system privileges, and roles granted to SYS_AUTH_OPERATOR_ROLE to the new role and drops SYS_AUTH_OPERATOR_ROLE.

```sql
ALTER ROLE SYS_AUTH_OPERATOR_ROLE
MIGRATE TO Operator_role
```

Usage

During the migration process:

- A new user-defined role is created.
- All of the system privileges currently granted to the migrating predefined role are automatically granted to the new user-defined role.
- All users and roles currently granted to the migrating predefined role are automatically granted to the new user-defined role.
• The compatibility role is dropped.

Since no role administrator was specified during the migration process, only global role administrators can manage the new role. Use the CREATE ROLE statement to add role administrators with appropriate administrative rights to the role.

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.

**Permissions**

Requires the MANAGE ROLES system privilege granted with administrative rights.

**ALTER USER Statement**

Changes user settings.

**Syntax**

Syntax 1 – Change the definition of a database user

```
ALTER USER user-name
| [ IDENTIFIED BY password ]
| [ LOGIN POLICY policy-name ]
| [ FORCE PASSWORD CHANGE { ON | OFF } ]
```

Syntax 2 – Refresh the Distinguished Name (DN) for an LDAP user

```
ALTER USER user-name
   REFRESH DN
```

Syntax 2 – Revert a user's login policy to the original values

```
ALTER USER user-name
   RESET LOGIN POLICY
```

Syntax 3 – Change a user's password when CHANGE_PASSWORD_DUAL_CONTROL is enabled in a user's login policy.

```
ALTER USER user-name
   IDENTIFIED [ FIRST | LAST ] BY password_part
```

**Parameters**

• `user-name` – name of the user. Multiple user names cannot be specified when issuing this command.

• `IDENTIFIED BY` – clause providing the password for the user. Clause not supported (ERROR) when CHANGE_PASSWORD_DUAL_CONTROL option is enabled in a user's login policy
• **IDENTIFIED[ FIRST | LAST | BY** – clause mandatory when
  CHANGE_PASSWORD_DUAL_CONTROL option is enabled in a target user’s login
  policy. FIRST | LAST keyword specifies the part of the dual password part being defined.
• **policy-name** – name of the login policy to assign the user. No change is made if the LOGIN
  POLICY clause is not specified.
• **FORCE PASSWORD CHANGE** – controls whether the user must specify a new
  password when he or she logs in. This setting overrides the
  PASSWORD_EXPIRY_ON_NEXT_LOGIN option setting in their policy.

  **Note:** This functionality is not currently implemented when logging in to Sybase Control
  Center. A user will not be prompted to change their password. He or she will be prompted,
  however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for
  example, using Interactive SQL).

• **RESET LOGIN POLICY** – reverts the settings of the user's login to the original values in
  the login policy. This usually clears all locks that are implicitly set due to the user
  exceeding the failed logins or exceeding the maximum number of days since the last login.
  When you reset a login policy, a user can access an account that has been locked for
  exceeding a login policy option limit such as MAX_FAILED_LOGIN_ATTEMPTS or
  MAX_DAYS_SINCE_LOGIN.

• **REFRESH DN** – clears the saved DN and timestamp for a user, which is used during
  LDAP authentication.

**Examples**

• **Example 1** – alters a user named SQLTester. The password is set to welcome. The
  SQLTester user is assigned to the Test1 login policy and the password does not expire on
  the next login:

  ```sql
  ALTER USER SQLTester
  IDENTIFIED BY welcome
  LOGIN POLICY Test1
  FORCE PASSWORD CHANGE OFF
  ```

• **Example 2** – clears the distinguished name (DN) and timestamp for a user named Mary
  used for LDAP authentication:

  ```sql
  ALTER USER Mary REFRESH DN
  ```

• **Example 3** – sets the password for user3 to PassPart1PassPart2. This assumes that user1
  and user2 have the CHANGE_PASSWORD system privilege and the
  change_password_dual_control option is enabled (ON) in the login policy for user3:

  User1 enters: ALTER USER user3 IDENTIFIED FIRST BY PassPart1
  User2 enters: ALTER USER user3 IDENTIFIED LAST BY PassPart2
  Once set, user3 logs on by entering the password PassPart1PassPart2.
**Usage**

User IDs and passwords cannot:

- Begin with white space, single quotes, or double quotes
- End with white space
- Contain semicolons

Passwords cannot exceed 255 characters.

If you set the `PASSWORD_EXPIRY_ON_NEXT_LOGIN` value to ON, the passwords of all users assigned to this login policy expire immediately when he or she next logs in. You can use the `ALTER USER` and `LOGIN POLICY` clauses to force users to change their passwords at the next login.

If the `CHANGE_PASSWORD_DUAL CONTROL` login policy option is disable (OFF) during the dual password change process:

- the target user will be unable to log in with the single password part already defined. The `ALTER USER` command must be reissued using single password control syntax.
- If the option is disabled after the dual password change process is complete, but before the target user logs in, there is no impact on the target user. The target user must log in using both password parts.

If the target user is already logged in when the dual password change process occurs, the user cannot change their password in the current session until both parts of the new password are set. Once the dual password change process is complete, the target user can use `GRANT CONNECT`, `ALTER USER`, `sp_password`, or `sp_iqpassword` to the password without first logging out. The prompt to enter the current password, use the new dual control password, not the password originally entered for the current session.

The `GRANT CONNECT` statement is not supported during for the dual password change process to set either password part. However, once the dual password change process is complete, the target user can use the `GRANT CONNECT`, `ALTER USER`, `sp_password`, or `sp_iqpassword` to change their password without first logging out.

As soon as both parts of the password are successfully specified by users with the `CHANGE PASSWORD` system privilege, the password for the target user is automatically expired. This forces the target user to change the password the next time he or she logs in.

The encryption algorithm used for hashing the user passwords is FIPS-certified encryption support:

- The DLL is called `dbfips10.dll`
- The HASH function accepts the algorithms: `SHA1_FIPS SHA256_FIPS`
- If the `-fips` server option is specified and an algorithm that is not FIPS-certified is given to the HASH function, the database server uses `SHA1_FIPS` instead of `SHA1, SHA256_FIPS`
instead of SHA256, and returns an error if MD5 is used (MD5 is not a FIPS-certified algorithm).
• If the -fips option is specified, the database server uses SHA256_FIPS for password hashing.

Standards
• SQL – Vendor extension to ISO/ANSI SQL grammar.
• Sybase – Not supported by Adaptive Server Enterprise.

Permissions
• To change own password – None required.
• To change the password of any user – Requires the CHANGE PASSWORD system privilege.
• To use the LOGIN POLICY, FORCE PASSWORD CHANGE, RESET LOGIN POLICY, or REFRESH DN clauses requires the MANAGE ANY USER system privilege.

CREATE LDAP SERVER Statement
Creates a new LDAP server configuration object for LDAP user authentication. Parameters defined during the creation of an LDAP server configuration object are stored in the ISYSLDAPSERVER (system view SYSLDAPSERVER) system table.

Syntax
CREATE LDAP SERVER ldapua-server-name
    [ ldapua-server-attribs ]
    [ WITH ACTIVATE ]

ldapua-server-attribs:
SEARCH DN
    URL { 'URL_string' | NULL }
    | ACCESS ACCOUNT { 'DN_string' | NULL }
    | IDENTIFIED BY { 'password>' | NULL }
    | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
| AUTHENTICATION URL { 'URL_string' | NULL }
| CONNECTION TIMEOUT timeout_value
| CONNECTION RETRIES retry_value
| TLS { ON | OFF }

Parameters
• URL – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP URL syntax before it is stored in the ISYSLDAPSERVER system table. The maximum size for this string is 1024 bytes.
• **ACCESS ACCOUNT** – a user created in the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect to the LDAP server. This user has permissions within the LDAP server to search for DNs by user ID in the locations specified by the SEARCH DN URL. The maximum size for this string is 1024 bytes.

• **IDENTIFIED BY** – provides the password associated with the ACCESS ACCOUNT user. The password is stored using symmetric encryption on disk. Use the value NULL to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.

• **IDENTIFIED BY ENCRYPTED** – configures the password associated with the ACCESS ACCOUNT distinguished name in an encrypted format. The binary value is the encrypted password and is stored on disk as is. Use the value NULL to clear the password and set it to none. The maximum size of the binary is 289 bytes. The encrypted key should be a valid varbinary value. Do not enclose the encrypted key in quotation marks.

• **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for `<URL_string>` and is validated for correct LDAP URL syntax before it is stored in `ISYSLDAPSERVER` system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.

• **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.

• **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase 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 for both DN searches and authentication. When set to ON, the TLS protocol is used and the URL would begin with "ldap://". When set to OFF (or not specified), Secure LDAP protocol is used and the URL begins with "ldaps://". When using the TLS protocol, specify the database security option TRUSTED_CERTIFICATES_FILE with a file name containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.

• **WITH ACTIVATE** – activates the LDAP server configuration object for immediate use upon creation. This permits the definition and activation of LDAP User Authentication in one statement. The LDAP server configuration object state changes to READY when WITH ACTIVATE is used.

### Examples

• **Example 1** – sets the search parameters, the authentication URL, and sets a three second timeout, and activates the server so it can begin authenticating users. It connects to the LDAP server without TLS or SECURE LDAP protocols.
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'
CREATE LDAP SERVER apps_primary
SEARCH DN
  URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'
  ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
  IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'
CONNECTION TIMEOUT 3000
WITH ACTIVATE

- **Example 2** – uses the same search parameters as example 1, but specifies “ldaps” so that a Secure LDAP connection is established with the LDAP server on host my_LDAPserver, port 636. Only LDAP clients using the Secure LDAP protocol may now connect on this port. The database security option TRUSTED_CERTIFICATE_FILE must be set with a file name containing the certificate of the certificate authority (CA) that signed the certificate used by the LDAP server at "ldaps://my_LDAPserver:636". During the handshake with the LDAP server, the certificate presented by the LDAP server is checked by the SAP Sybase IQ server (the LDAP client) to ensure that it is signed by one of the certificates listed in the file. This establishes trust by the client that the server is who it says it is. The ACCESS ACCOUNT and IDENTIFIED BY parameters establish trust by the LDAP server that the client is who it says it is.

**Note:** The TLS parameter must be OFF when Secure LDAP is used instead of TLS protocol.

SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'
SET OPTION PUBLIC.trusted_certificates_file = '/mycompany/shared/trusted.txt'
CREATE LDAP SERVER secure_primary
SEARCH DN
  URL 'ldaps://my_LDAPserver:636/dc=MyCompany,dc=com??sub?cn=*'
  ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
  IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldaps://my_LDAPserver:636/'
CONNECTION TIMEOUT 3000
TLS OFF
WITH ACTIVATE

- **Example 3** – establishes the TLS protocol on port 389. It also requires database security option TRUSTED_CERTIFICATE_FILE to be set with a file name and provides the same type of security as example 2. In this example, the TLS protocol is ON to facilitate wider support by LDAP server vendors.

**Note:** Check the requirements of all your LDAP servers when deciding how to configure Secure LDAP or TLS for an SAP Sybase IQ server.

SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'
SET OPTION PUBLIC.trusted_certificates_file = '/mycompany/shared/trusted.txt'
CREATE LDAP SERVER tls_primary
SEARCH DN
  URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'
Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ANY LDAP SERVER system privilege.

CREATE LOGIN POLICY Statement

Creates a login policy in the database.

Syntax

CREATE LOGIN POLICY  policy-name  
AUTO_UNLOCK_TIME=0 – UNLIMITED  
|  DEFAULT_LOGICAL_SERVER=[logical_server_name | ALL | AUTO | COORDINATOR | NONE | OPEN | SERVER]  
|  CHANGE_PASSWORD_DUAL_CONTROL=[ON | OFF]  
|  LOCKED=[ON | OFF]  
|  MAX_CONNECTIONS=0 – 2147483647  
|  MAX_DAYS_SINCE_LOGIN=0 – 2147483647  
|  MAX_FAILED_LOGIN_ATTEMPTS=0 – 2147483647  
|  MAX_NON_DBA_CONNECTIONS=0 – 2147483647  
|  PASSWORD_EXPIRY_ON_NEXT_LOGIN=[ON | OFF]  
|  PASSWORD_GRACE_TIME=0 – 2147483647  
|  PASSWORD_LIFE_TIME=0 – 2147483647  
|  ROOT_AUTO_UNLOCK_TIME=0 – UNLIMITED  
|  LDAP_PRIMARY_SERVER=server_name  
|  LDAP_SECONDARY_SERVER=server_name  
|  LDAP_AUTO_FAILBACK_PERIOD=0 – 2147483647  
|  LDAP_FAILOVER_TO_STD=[ON | OFF]  
|  LDAP_REFRESH_DN=NOW

Applies to

Simplex and multiplex.

Examples

• Example 1 – creates the Test1 login policy. This login policy has an unlimited password life and allows the user a maximum of five attempts to enter a correct password before the account is locked.
CREATE LOGIN POLICY Test1
password_life_time=UNLIMITED
max_failed_login_attempts=5;

Usage
If you do not specify a login policy option, the value from the root login policy is applied.

Permissions
Requires MANAGE ANY LOGIN POLICY system privilege.

The following system privileges can override the noted login policy options:

<table>
<thead>
<tr>
<th>Exception System Privilege</th>
<th>Login Policy Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>SERVER OPERATOR or DROP CONNECTION system privilege</td>
<td>MAX_NON_DBA_CONNS</td>
</tr>
<tr>
<td></td>
<td>MAX_CONNECTIONS</td>
</tr>
<tr>
<td>MANAGE ANY USER system privilege</td>
<td>LOCKED</td>
</tr>
<tr>
<td></td>
<td>MAX_DAYS_SINCE_LOGIN</td>
</tr>
</tbody>
</table>

Login Policy Options
Available options for root and user-defined login policies.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTO_UNLOCK_TIME</td>
<td>The time period after which locked accounts not granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined in any login policy, including the root login policy.</td>
</tr>
<tr>
<td></td>
<td>• Values – 0 – unlimited</td>
</tr>
<tr>
<td></td>
<td>• Initial value for Root policy – Unlimited</td>
</tr>
<tr>
<td></td>
<td>• Applies to – All users not granted the MANAGE ANY USER system privilege.</td>
</tr>
<tr>
<td>CHANGE_PASSWORD_DUAL_CONTROL</td>
<td>Requires input from two users, each granted the CHANGE PASSWORD system privilege, to change the password of another user.</td>
</tr>
<tr>
<td></td>
<td>• Values – ON, OFF</td>
</tr>
<tr>
<td></td>
<td>• Initial value for Root policy – OFF</td>
</tr>
<tr>
<td></td>
<td>• Applies to – All users.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| DEFAULT_LOGICAL_SERVER   | If the connection string specifies no logical server, the user connects to the DEFAULT_LOGICAL_SERVER setting specified in the user's login policy.  
  • Values –  
  • Name of an existing user-defined logical server  
  • ALL – allows access to all logical servers.  
  • 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.  
  • Initial value for Root policy – 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  
  • Initial value for Root policy – 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  
  • Initial value for Root policy – Unlimited  
  • Applies to – All users except those with the SERVER OPERATOR or DROP CONNECTION system privilege.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| MAX_DAYS_SINCE_LOGIN                        | The maximum number of days that can elapse between two successive logins by the same user.  
  • **Values** – 0 – 2147483647  
  • **Initial value for Root policy** – 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 into the user account before the account is locked.  
  • **Values** – 0 – 2147483647  
  • **Initial value for Root policy** – Unlimited  
  • **Applies to** – All users.                                                                                                                                                                                                                                                     |
| 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  
  • **Initial value for Root policy** – Unlimited  
  • **Applies to** – All users except those with the SERVER OPERATOR or DROP CONNECTION privilege.                                                                                                                                                                                                                               |
| PASSWORD_EXPIRY_ON_NEXT_LOGIN               | If set ON, the user's password expires at the next login.  
  • **Values** – ON, OFF  
  • **Initial value for Root policy** – OFF  
  • **Applies to** – All users.                                                                                                                                                                                                                                                     |
| Note: This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).                                                                 |                                                                                                                                                                                                                                                                             |
| 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  
  • **Initial value for Root policy** – 0  
  • **Applies to** – All users.                                                                                                                                                                                                                                                     |
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| PASS-WORD_LIFE_TIME | The maximum number of days before a password must be changed.  
  - **Values** – 0 – 2147483647  
  - **Initial value for Root policy** – Unlimited  
  - **Applies to** – All users. |
| ROOT_AUTO_UNLOCK_TIME | The time period after which locked accounts granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined only in the root login policy.  
  - **Values** – 0 – unlimited  
  - **Initial value for Root policy** – 15  
  - **Applies to** – All users granted the MANAGE ANY USER system privilege. |

### LDAP Login Policy Options

Available login policy options for LDAP user authentication

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| LDAP_PRIMARY_SERVER | Specifies the name of the primary LDAP server.  
  - **Values** – n/a  
  - **Initial value for Root policy** – None  
  - **Applies to** – All users. |
| LDAP_SECONDARY_SERVER | Specifies the name of the secondary LDAP server.  
  - **Values** – n/a  
  - **Initial value for ROOT policy** – None  
  - **Applies to** – All users. |
| LDAP_AUTO_FAILBACK_PERIOD | Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.  
  - **Values** – 0 - 2147483647  
  - **Initial value for ROOT policy** – 15 minutes  
  - **Applies to** – All users. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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
• Initial value for ROOT policy – ON
• 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 the option ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the user_dn value 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
• Applies to – All users.                                                                                                                                                                                                                                                                              |

**Multiplex Login Policy Configuration**
Configure login policies for multiplex servers.

**Example**

This example overrides the login policy settings on a logical server, increasing the maximum number of connections on logical server ls1:

```
ALTER LOGIN POLICY lp1 max_connections=20 LOGICAL SERVER ls1;
```

**Usage**

Applies only to multiplex.

Any login management commands you execute on any multiplex server automatically propagate to all servers in the multiplex. For best performance, execute these commands, or any DDL, on the coordinator.

An override at the logical server level override means that a particular login policy option has different settings for different logical servers. **SYS.ISYSIQLSLOGINPOLICYOPTION** stores login policy option values for logical-server override. For each logical-server override
of a login policy option, a corresponding row exists in ISYSIQSLIGNSLOGINPOLICYOPTION.

**CREATE ROLE Statement**

Creates a new role, extends an existing user to act as a role, or manages role administrators on a role.

**Syntax**

```
CREATE [ OR REPLACE ] ROLE role_name | FOR USER userID
[ [ WITH ADMIN [ ONLY ] ] admin_name [, ... ] ]
```

**Parameters**

- **role_name** – unless you are using the OR REPLACE clause, `role_name` cannot already exist in the database.
- **OR REPLACE** – `role_name` must already exist in the database. All current administrators are replaced by those specified in `admin_name [,...]` clause.
- **FOR USER** – when using the FOR USER clause without the OR REPLACE, `userID` must be the name of an existing user that currently does not have the ability to act as a role.
- **admin_name** – list of users to be designated administrators of the role.
- **WITH ADMIN** – each `admin_name` specified is granted administrative privileges over the role in addition to all underlying system privileges.
- **WITH ADMIN ONLY** – each `admin_name` specified is granted administrative privileges only over the role, not the underlying system privileges.

**Note:** If you do not specify an ADMIN clause, the default WITH ADMIN ONLY clause is used and the default administrator is the global roles administrator.

**Examples**

- **Example 1** – creates the role `Sales` as a standalone role. Since no role administrator is specified, any user with the global role administrator is the default administrator of this role.
  ```sql
  CREATE ROLE Sales
  ```

- **Example 2** – extends the role for existing user `Jane`.
  ```sql
  CREATE OR REPLACE ROLE FOR USER Jane
  ```

- **Example 3** – creates the role `Finance` with `Mary` and `Jeff` as role administrators with administrative rights only to the role.
  ```sql
  CREATE ROLE Finance
  WITH ADMIN ONLY Mary, Jeff
  ```
• **Example 4** – if Finance is an existing role with Mary and Jeff with role administrators, this statement replaces Mary and Jeff with Bob and Sarah as role administrators, this time with administrative rights to the role.

```sql
CREATE OR REPLACE ROLE Finance
WITH ADMIN Bob, Sarah
```

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.

**Permissions**

• Create a new role – Requires the MANAGE ROLES system privilege.
• OR REPLACE clause – Requires the MANAGE ROLES system privilege along with administrative rights over the role being replaced.

**CREATE USER Statement**

Creates a user.

**Syntax**

```sql
CREATE USER user-name [ IDENTIFIED BY password ]
[ LOGIN POLICY policy-name ]
[ FORCE PASSWORD CHANGE { ON | OFF } ]
```

**Parameters**

• **user-name** – name of the user.
• **IDENTIFIED BY** – provides the password for the user.
• **policy-name** – name of the login policy to assign the user. No change is made if you do not specify a login policy.
• **FORCE PASSWORD CHANGE** – controls whether the user must specify a new password upon logging in. This setting overrides the PASSWORD_EXPIRY_ON_NEXT_LOGIN option setting in the user's login policy.

**Note:** This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).

• **password** – You do not have to specify a password for the user. A user without a password cannot connect to the database. This is useful if you are creating a role and do not want anyone to connect to the database using the role user ID. A user ID must be a valid identifier.

User IDs and passwords cannot:
Appendix: SQL Reference

- Begin with white space, single quotes, or double quotes
- End with white space
- Contain semicolons

A password can be either a valid identifier, or a string (maximum 255 characters) placed in single quotes. Passwords are case-sensitive. The password should be composed of 7-bit ASCII characters, as other characters may not work correctly if the database server cannot convert them from the client's character set to UTF-8.

You can use the VERIFY_PASSWORD_FUNCTION option to specify a function to implement password rules (for example, passwords must include at least one digit). If you do use a password verification function, you cannot specify more than one user ID and password in the GRANT CONNECT statement.

The encryption algorithm used for hashing the user passwords is FIPS-certified encryption support:

- The DLL is called dbfips10.dll
- The HASH function accepts the algorithms: SHA1_FIPS SHA256_FIPS
- If the -fips server option is specified and an algorithm that is not FIPS-certified is given to the HASH function, the database server uses SHA1_FIPS instead of SHA1, SHA256_FIPS instead of SHA256, and returns an error if MD5 is used (MD5 is not a FIPS-certified algorithm).
- If the -fips option is specified, the database server uses SHA256_FIPS for password hashing.

Examples

- Example 1 – creates a user named SQLTester with the password welcome. The SQLTester user is assigned to the Test1 login policy and the password expires on the next login:

```sql
CREATE USER SQLTester IDENTIFIED BY welcome
LOGIN POLICY Test1
FORCE PASSWORD CHANGE ON;
```

Standards

- SQL – Vendor extension to ISO/ANSI SQL grammar.
- Sybase – Not supported by Adaptive Server Enterprise.

Permissions

Requires the MANAGE ANY USER system privilege.
DROP LDAP SERVER Statement

Removes the named LDAP server configuration object from the SYSLDAPSERVER system view after verifying that the LDAP server configuration object is not in a READY or ACTIVE state.

The DROP LDAP SERVER statement fails when it is issued against an LDAP server configuration object that is in a READY or ACTIVE state. This ensures that an LDAP server configuration object in active use cannot be accidentally dropped. The DROP LDAP SERVER statement also fails if a login policy exists with a reference to the LDAP server configuration object.

Syntax

```
DROP LDAP SERVER <ldapua-server-name>
[ WITH DROP ALL REFERENCES ] [ WITH SUSPEND ]
```

Parameters

- **WITH DROP ALL REFERENCES** – allows the removal of an LDAP server configuration object from service that has a reference in a login policy.
- **WITH SUSPEND** – allows an LDAP server configuration object to be dropped even if in a READY or ACTIVE state.

Examples

- **Example 1** – assuming that references to the LDAP server configuration object have been removed from all login policies, the following two sets of commands are equivalent. Using the WITH DROP ALL REFERENCES and WITH SUSPEND parameters eliminates the need to execute an ALTER LDAP SERVER statement before the DROP LDAP SERVER statement:

  ```
  DROP LDAP SERVER ldapserver1 WITH DROP ALL REFERENCES WITH SUSPEND
  ```

  is equivalent to

  ```
  ALTER LDAP SERVER ldapserver1 WITH SUSPEND
  DROP LDAP SERVER ldapserver1 WITH DROP ALL REFERENCES
  ```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ANY LDAP SERVER system privilege.
DROP LOGIN POLICY Statement

Removes a login policy from the database.

Syntax

DROP LOGIN POLICY  policy-name

Examples

• Example 1 – Create and then delete the Test11 login policy:

  CREATE LOGIN POLICY Test11;
  DROP LOGIN POLICY Test11

Usage

A DROP LOGIN POLICY statement fails if you attempt to drop a policy that is assigned to a user. You can use either the ALTER USER statement to change the policy assignment of the user or DROP USER to drop the user.

Permissions

Requires the MANAGE ANY LOGIN POLICY system privilege.

DROP ROLE Statement

Removes a user-defined role from the database or converts a user-extended role to a regular user.

Syntax

DROP ROLE [ FROM USER ]  role_name
[ WITH REVOKE ]
[ WITH DROP OBJECTS ]

Parameters

• role_name – must be the name of a role that already exists in the database.
• FROM USER – required to convert a user-extended role back to a regular user. The role_name must exist in the database.
• WITH REVOKE – required when dropping a standalone or user-extended role to which users have been granted the underlying system privileges of the role with either the WITH ADMIN OPTION or WITH NO ADMIN OPTION clause.
• WITH DROP OBJECTS – required when dropping a standalone or user-extended role that owns objects.
Examples

- **Example 1** – converts a user-extended role named *Joe* that does not have its underlying system privileges granted to any user back to a regular user.

  \[
  \text{DROP ROLE FROM USER Joe}
  \]

- **Example 2** – drops a user-extended role named *Jack* that does not have its underlying system privileges granted to any user from the database.

  \[
  \text{DROP ROLE Jack}
  \]

- **Example 3** – converts a user-extended role named *Sam* that has underlying system privileges granted to users back to a regular role.

  \[
  \text{DROP ROLE FROM USER Sam WITH REVOKE}
  \]

- **Example 4** – drops a standalone role named *Sales1* that does not own objects, nor has its underlying system privileges granted to any user from the database.

  \[
  \text{DROP ROLE Sales1}
  \]

- **Example 5** – drops a standalone role named *Sales2* that does not own objects, and has its underlying system privileges granted to users from the database.

  \[
  \text{DROP ROLE Sales2 WITH REVOKE}
  \]

- **Example 6** – converts a user-extended role named *Marketing1* that owns objects, but does not have its underlying system privileges granted to users back to a regular user.

  \[
  \text{DROP ROLE FROM USER Marketing1 WITH DROP OBJECTS}
  \]

- **Example 7** – drops a standalone role named *Marketing2* that owns objects, and has its underlying system privileges granted to users from the database.

  \[
  \text{DROP ROLE Marketing1 WITH REVOKE WITH DROP OBJECTS}
  \]

Usage

A user-defined role can be dropped from the database or converted back to a regular user at any time as long as all dependent roles left meet the minimum required number of administrative users with active passwords.

Include the FROM USER clause when dropping a user-extended role to convert it back to act as a regular user rather than remove it from the database. The user retains any login privileges, system privileges, and roles granted to the user-extended role and becomes the owner of any objects owned by the user-extended role. Any users granted to the user-extended are immediately revoked.
Standards
ANSI SQL – Compliance level: Transact-SQL extension.

Permissions
• Requires administrative rights over the role being dropped.
• If the role being dropped owns objects, none are in use by any user in any session at the time the DROP statement is executed.

DROP USER Statement
Removes a user.

Syntax
DROP USER  user-name

Parameters
•  user-name – name of the user to remove.

Examples
•  Example 1 – drops the user SQLTester from the database:
  DROP USER SQLTester

Standards
• SQL – ISO/ANSI SQL compliant.
• Sybase – Not supported by Adaptive Server Enterprise.

Permissions
Requires the MANAGE ANY USER system privilege.

Note: When dropping a user, any objects owned by this user and any permissions granted by this user will be removed.

GRANT CHANGE PASSWORD Statement
Allows users to manage passwords for other users and administer the CHANGE PASSWORD system privilege.

A user can be granted the ability to manage the password of any user in the database (ANY) or 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 only be granted when using the ANY clause.
**Syntax**

GRANT CHANGE PASSWORD (target_user_list | ANY | ANY WITH ROLES
target_role_list) TO userID [, ...]
[ WITH ADMIN [ONLY] OPTION | WITH NO ADMIN OPTION]

**Parameters**

- **target_user_list** – users the grantee has the potential to impersonate. The list must consist of existing users or user-extended roles with login passwords. Separate the userIDs in the list with commas.
- **ANY** – all database users with login passwords become potential target users to manage passwords for each grantee.
- **ANY WITH ROLES target_role_list** – list of target roles for each grantee. Any users who are granted any of the target roles become potential target users for each grantee. The target_role_list must consist of existing roles and the users who are granted said roles must consist of database users with login passwords. Use commas to separate multiple userIDs.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **WITH ADMIN OPTION** – (valid with the **ANY** clause only) The user can both manage passwords and grant the CHANGE PASSWORD system privilege to another user.
- **WITH ADMIN ONLY OPTION** – (valid with the **ANY** clause only) The user can grant the CHANGE PASSWORD system privilege to another user, but cannot manage passwords of other users.
- **WITH NO ADMIN OPTION** – the user can manage passwords, but cannot grant the CHANGE PASSWORD system privilege to another user.

**Examples**

- **Example 1** – grants Sally and Laurel the ability to manage the password of Bob, Sam, and Peter.

  GRANT CHANGE PASSWORD (Bob, Sam, Peter) TO (Sally, Laurel)

- **Example 2** – grants Mary the right to grant the CHANGE PASSWORD system privilege to any user in the database. However, since the system privilege is granted with the WITH ADMIN ONLY OPTION clause, Mary cannot manage the password of any other user.

  GRANT CHANGE PASSWORD (ANY) TO Mary WITH ADMIN ONLY OPTION

- **Example 3** – grants Steve and Joe the ability to manage the password of any member of Role1 or Role2.

  GRANT CHANGE PASSWORD (ANY WITH ROLES Role1, Role2) TO Steve, Joe

**Usage**

If no clause is specified, **ANY** is used by default.
If no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

By default, the CHANGE PASSWORD system privilege is granted to the SYS_AUTH_SA_ROLE compatibility role with the WITH NO ADMIN OPTION clause and to the SYS_AUTH_SSO_ROLE compatibility role with the ADMIN ONLY OPTION clause, if they exist.

Standards
ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

- Requires the CHANGE PASSWORD system privilege granted with administrative rights.
- Each target user specified (target_users_list) is 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.

GRANT CONNECT Statement
Grants CONNECT privilege to a user.

GRANT CONNECT can be used to create a new user or also be used by any user to change their own password.

Tip: Use the CREATE USER statement rather than the GRANT CONNECT statement to create users.

Syntax

```
GRANT CONNECT
TO userID [, ...]
IDENTIFIED BY password [, ...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – creates two new users for the database named Laurel and Hardy.

  ```
  GRANT CONNECT TO Laurel, Hardy
  IDENTIFIED BY Stan, Ollie
  ```

- **Example 2** – creates user Jane with no password.
GRANT CONNECT TO Jane

- **Example 3** – changes the password for Bob to newpassword.

GRANT CONNECT TO Bob IDENTIFIED BY newpassword

**Usage**

The same command can be used to both create a new user or change the password of an existing user. If you inadvertently enter the user ID of an existing user when you are trying to add a new user, you are actually changing the password of the existing user. You do not receive a warning because this behavior is considered normal.

The stored procedures `sp_addlogin` and `sp_adduser` can also be used to add users. These procedures display an error if you try to add an existing user ID.

**Note:** Use system procedures, not GRANT and REVOKE, to add and remove user IDs.

A user without a password cannot connect to the database. This is useful when you are creating groups and you do not want anyone to connect to the role user ID. To create a user without a password, do not include the IDENTIFIED BY clause.

When specifying a password, it must be a valid identifier. Passwords have a maximum length of 255 bytes. If the VERIFY_PASSWORD_FUNCTION database option is set to a value other than the empty string, the GRANT CONNECT TO statement calls the function identified by the option value. The function returns NULL to indicate that the password conforms to rules. If the VERIFY_PASSWORD_FUNCTION option is set, you can specify only one userid and password with the GRANT CONNECT statement.

Invalid names for database user IDs and passwords include those that:

- Begin with white space or single or double quotes
- End with white space
- Contain semicolons

**Standards**

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – The security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

**Permissions**

- If you are creating a new user, you must have the MANAGE ANY USER system privilege.
- Any user can change his or her own password.
- If you are changing another user’s password, you must have the CHANGE PASSWORD system privilege.
Note: If you are changing another user’s password, the other user cannot be connected to the database.

See also
• CREATE USER Statement on page 255

GRANT CREATE Statement
Grants CREATE privilege on a specified dbspace to the specified users and roles.

Syntax
GRANT CREATE
ON dbspace_name
TO userID [, ...]

Parameters
• userID – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples
• Example 1 – grants users Lawrence and Swift CREATE privilege on dbspace DspHist.
  GRANT CREATE ON DspHist
  TO LAWRENCE, SWIFT
• Example 2 – grants CREATE privilege on dbspace DspHist to users Fiona and Ciaran.
  GRANT CREATE ON DspHist TO Fiona, Ciaran

Standards
• SQL – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
• Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions
Requires the MANAGE ANY DBSPACE system privilege.
**GRANT EXECUTE Statement**

Grants EXECUTE privilege on a procedure or user-defined function.

**Syntax**

```sql
GRANT EXECUTE
   ON [ owner.] {procedure-name | user-defined-function-name } 
   TO userID [, ...]
```

**Parameters**

- `userID` – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

**Standards**

- SQL – syntax is a Persistent Stored Module feature.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

**Permissions**

Requires one of:

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

**GRANT Object-Level Privilege Statement**

Grants database object-level privileges to a user or role.

**Syntax**

```sql
GRANT object-level-privilege [, ...]
   ON [ owner.]object-name 
   TO userID [, ...] 
   [ WITH GRANT OPTION ]
```

**object-level-privilege:**

- `ALL [ PRIVILEGES ]`
- `ALTER`
- `DELETE`
- `INSERT`
- `REFERENCES [ ( column-name [, ...] ) ]`
- `SELECT [ ( column-name [, ...] ) ]`
- `UPDATE [ ( column-name, ... ) ]`
- `LOAD`
- `TRUNCATE`
Parameters

- **userID** – must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.

Usage

Grants privileges on individual tables or views. You can list the table privileges together, or specify **ALL** to grant all privileges at once. If you specify the **WITH GRANT OPTION** clause, the named user ID is also given privileges to grant the same privileges to other user IDs.

- **ALL** – grants all privileges to users
- **ALTER** – users can alter this table with the **ALTER TABLE** statement. This privilege is not allowed for views.
- **DELETE** – users can delete rows from this table or view.
- **INSERT** – users can insert rows into the named table or view.
- **LOAD** – users can load data into the named table or view.

These server switches may impact a user's ability to execute the LOAD command:

- **-gl NONE** – no one can execute the LOAD or UNLOAD command on a table.
- **-gl ALL** – users with ALTER ANY TABLE or LOAD ANY TABLE system privilege can execute LOAD command on any table. Table owners or users with ALTER or LOAD privilege on a given table can execute LOAD command on the table. Users with SELECT ANY TABLE system privilege or SELECT privilege on a given table can execute the UNLOAD command.
- **-gl DBA** – users with ALTER ANY TABLE or LOAD ANY TABLE system privilege can execute the LOAD command on any table.
- **REFERENCES** – users can create indexes on the named tables, and foreign keys that reference the named tables. If column names are specified, then users can reference only those columns. REFERENCES privileges on columns cannot be granted for views, only for tables.
- **SELECT** – users can look at information in this view or table. If column names are specified, then the users can look at only those columns. SELECT permissions on columns cannot be granted for views, only for tables.
- **TRUNCATE** – users can truncate the named table or view.
- **UPDATE** – users can update rows in this view or table. If column names are specified, users can update only those columns. UPDATE privileges on columns cannot be granted for views, only for tables. To update a table, users must have both SELECT and UPDATE privilege on the table.

Standards

- SQL – Syntax is an entry-level feature.
- Sybase – Syntax is supported in Adaptive Server Enterprise.
Permissions
Requires one of:

- MANAGE ANY OBJECT PRIVILEGE system privilege
- You have been granted the specific object privilege with the WITH GRANT OPTION clause on the table.
- You own of the table.

GRANT ROLE Statement
Grants roles to users or other roles, with or without administrative rights.

Syntax
GRANT ROLE  role_name [, ...]
TO  grantee [, ...]
[[ {WITH NO ADMIN | WITH ADMIN [ ONLY ]} OPTION ]
[ WITH NO SYSTEM PRIVILEGE INHERITANCE ]

role_name:
dbo†††
| diagnostics†††
PUBLIC†††
rs_systabgroup†††
SA_DEBUG†††
SYS†††
SYS_AUTH_SA_ROLE
SYS_AUTH_SSO_ROLE
SYS_AUTH_DBROLE††
SYS_AUTH_RESOURCE_ROLE†
SYS_AUTH_BACKUP_ROLE†
SYS_AUTH_VALIDATE_ROLE†
SYS_AUTH_WRITEFILE_ROLE
SYS_AUTH_WRITEFILECLIENT_ROLE
SYS_AUTH_READFILE_ROLE
SYS_AUTH_READFILECLIENT_ROLE
SYS_AUTH_PROFILE_ROLE
SYS_AUTH_USER_ADMIN_ROLE
SYS_AUTH_SPACE_ADMIN_ROLE
SYS_AUTH_MULTIPLEX_ADMIN_ROLE
SYS_AUTH_OPERATOR_ROLE
SYS_AUTH_PERMS_ADMIN_ROLE
SYS_REPLICATION_ADMIN_ROLE†††
SYS_RUN_REPLICATION_ROLE†††
SYS_SPATIAL_ADMIN_ROLE†††
<user-defined role name>

- The WITH NO SYSTEM PRIVILEGE INHERITANCE clause can be used when granting select compatibility roles to other roles. It prevents automatic inheritance of the compatibility role's underlying system privileges by members of the role. When granted to user-extended roles, the WITH NO SYSTEM PRIVILEGE INHERITANCE clause
applies to members of the role only. The user acting as a role automatically inherits the underlying system privileges regardless of the clause.

- The WITH NO ADMIN OPTION WITH NO SYSTEM PRIVILEGE INHERITANCE and WITH NO SYSTEM PRIVILEGE INHERITANCE clauses are semantically equivalent.

- ††The WITH ADMIN OPTION or WITH ADMIN ONLY clauses can not be specified in combination with the WITH NO SYSTEM PRIVILEGE INHERITANCE clause when granting the SYS_AUTH_BACKUP_ROLE, SYS_AUTH_RESOURCE_ROLE, or SYS_AUTH_VALIDATE_ROLE roles.

- †††The WITH ADMIN OPTION clause can only be specified in combination with the WITH NO SYSTEM PRIVILEGE INHERITANCE clause when granting the SYS_AUTH_DBA_ROLE or SYS_RUN_REPLICATION_ROLE roles.

- ††††The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for system roles.

Parameters

- role_name – must already exist in the database. Separate multiple role names with commas.

- grantee – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

- WITH NO ADMIN OPTION – each grantee is granted the underlying system privileges of each role_name, but cannot grant role_name to another user.

- WITH ADMIN ONLY OPTION – each userID is granted administrative privileges over each role_name, but not the underlying system privileges of role_name.

- WITH ADMIN OPTION – each userID is granted the underlying system privileges of each role_name, along with the ability to grant role_name to another user.

- WITH NO SYSTEM PRIVILEGE INHERITANCE – the underlying system privileges of the granting role are not inherited by the members of the receiving role. However, if the receiving role is a user-extended role, the underlying system privileges are granted to the extended user.

Note: Use of the WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clause allows the grantee to grant or revoke the role, but does not allow the grantee to drop the role.

Examples

- Example 1 – grants Sales_Role to Sally, with administrative privileges, which means she can grant or revoke Sales_Role to other users as well as perform any authorized tasks granted by the role.

  GRANT ROLE Sales_Role TO Sally WITH ADMIN OPTION

- Example 2 – grants the compatibility role SYS_AUTH_PROFILE_ROLE to the role Sales_Admin with no administrative rights. Sales_Admin is a standalone role and
Mary and Peter have been granted Sales_Admin. Since SYS_AUTH_PROFILE_ROLE is an inheritable compatibility role, Mary and Peter are granted the underlying system privileges of Sales_Role. Since the role is granted with no administrative rights, they cannot grant or revoke the role.

```
GRANT ROLE SYS_AUTH_PROFILE_ROLE TO Sales_Role WITH NO ADMIN OPTION
```

- **Example 3** – grants the compatibility role SYS_AUTH_BACKUP_ROLE to Tom with no administrative rights. Tom is a user-extended role to which Betty and Laurel have been granted. Since SYS_AUTH_BACKUP_ROLE is a non-inheritable compatibility role, the underlying system privileges of the role are not granted to Betty and Laurel. However, since Tom is an extended user, the underlying system privileges are granted directly to Tom.

```
GRANT ROLE SYS_AUTH_BACKUP_ROLE TO Tom
WITH NO SYSTEM PRIVILEGE INHERITANCE
```

**Usage**

- By default, if no administrative clause is specified in the grant statement, each compatibility role is granted with these default administrative rights:
<table>
<thead>
<tr>
<th>WITH ADMIN OPTION</th>
<th>WITH ADMIN ONLY OPTION</th>
<th>WITH NO ADMIN OPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_AUTH_SA_ROLE</td>
<td>SYS_AUTH_DBA_ROLE</td>
<td>SYS_AUTH_RESOURCE_ROLE</td>
</tr>
<tr>
<td>SYS_AUTH_SSO_ROLE</td>
<td></td>
<td>SYS_AUTH_BAKCUP_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_VALIDATE_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_WRITEFILE_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_WRITEFILECLIENT_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_READFILE_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_READFILECLIENT_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_PROFILE_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_USER_ADMIN_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_SPACE_ADMIN_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_MULTIPLEX_ADMIN_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_AUTH_OPERATOR_ROLE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA_DEBUG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYS_RUN_REPLICATION_ROLE</td>
</tr>
</tbody>
</table>

- The SYS_AUTH_PERMS_ADMIN_ROLE role grants these underlying roles with these default administrative rights:
WITH ADMIN OPTION | WITH NO ADMIN OPTION
--- | ---
SYS_AUTH_BACKUP_ROLE | MANAGE ROLES
SYS_AUTH_OPERATOR_ROLE | MANAGE ANY OBJECT PRIVILEGE
SYS_AUTH_USER_ADMIN_ROLE | CHANGE PASSWORD
SYS_AUTH_SPACE_ADMIN_ROLE | 
SYS_AUTH_MULTIPLEX_ADMIN_ROLE | 
SYS_AUTH_RESOURCE_ROLE | 
SYS_AUTH_VALIDATE_ROLE | 
SYS_AUTH_PROFILE_ROLE | 
SYS_AUTH_WRITEFILE_ROLE | 
SYS_AUTH_WRITEFILECLIENT_ROLE | 
SYS_AUTH_READFILE_ROLE | 
SYS_AUTH_READFILECLIENT_ROLE | 

Standards

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions

- Requires MANAGE ROLES system privilege to grant these system roles:
  - dbo
  - diagnostics
  - PUBLIC
  - rs_systabgroup
  - SA_DEBUG SYS
  - SYS
  - SYS_REPLICATION_ADMIN_ROLE
  - SYS_RUN_REPLICATION_ROLE
  - SYS_SPATIAL_ADMIN_ROLE
- Requires administrative privilege over the role to grant these roles:
  - SYS_AUTH_SA_ROLE
  - SYS_AUTH_SSO_ROLE
  - SYS_AUTH_DBA_ROLE
  - SYS_AUTH_RESOURCE_ROLE
  - SYS_AUTH_BACKUP_ROLE
Appendix: SQL Reference

- SYS_AUTH_VALIDATE_ROLE
- SYS_AUTH_WRITEFILE_ROLE
- SYS_AUTH_WRITEFILECLIENT_ROLE
- SYS_AUTH_READFILE_ROLE
- SYS_AUTH_READFILECLIENT_ROLE
- SYS_AUTH_PROFILE_ROLE
- SYS_AUTH_USER_ADMIN_ROLE
- SYS_AUTH_SPACE_ADMIN_ROLE
- SYS_AUTH_MULTIPLEX_ADMIN_ROLE
- SYS_AUTH_OPERATOR_ROLE
- SYS_AUTH_PERMS_ADMIN_ROLE
- <user-defined role name>

**GRANT SET USER Statement**

Grants the ability for one user to impersonate another user and to administer the SET USER system privilege.

A user can be granted the ability to impersonate any user in the database (ANY) or 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 only be granted when using the ANY clause.

**Syntax**

```
GRANT SET USER (target_users_list | ANY | ANY WITH ROLES target_roles_list) TO userID [, ...] [ WITH ADMIN [ONLY] OPTION | WITH NO ADMIN OPTION]
```

**Parameters**

- **target_users_list** – users the grantee has the potential to impersonate. The list must consist of existing users or user-extended roles with login passwords. Separate the userIDs in the list with commas.
- **ANY** – all database users with login passwords become potential target users for impersonation for each grantee.
- **ANY WITH ROLES target_roles_list** – list of target roles for each grantee. Any users who are granted any of the target roles become potential target users for each grantee. The target_role_list must consist of existing roles and the users who are granted said roles must consist of database users with login passwords. Use commas to separate multiple userIDs. There are two restrictions when using this method.
  - Only those users who have been granted a subset of the target_role_list can be impersonated by a grantee.
• Any user being impersonated must have exactly the exact subset of target_role_list; no additional roles are allowed.

• **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

• **WITH ADMIN OPTION** – (valid in conjunction with the **ANY** clause only) The user can both issue the SETUSER command to impersonate another user and grant the SET USER system privilege to another user.

• **WITH ADMIN ONLY OPTION** – (valid in conjunction with the **ANY** clause only) The user can grant the SET USER system privilege to another user, but cannot issue the SETUSER command to impersonate another user.

• **WITH NO ADMIN OPTION** – the user can issue the SETUSER command to impersonate another user, but cannot grant the SET USER system privilege to another user.

**Examples**

• **Example 1** – grants Sally and Laurel the ability to impersonate Bob, Sam, and Peter.

  ```sql
  GRANT SET USER (Bob, Sam, Peter) TO (Sally, Laurel)
  ```

• **Example 2** – grants Mary the right to grant the SET USER system privilege to any user in the database. However, since the system privilege is granted with the WITH ADMIN ONLY OPTION clause, Mary cannot impersonate any other user.

  ```sql
  GRANT SET USER (ANY) TO Mary WITH ADMIN ONLY OPTION
  ```

• **Example 3** – grants Steve and Joe the ability to impersonate any member of Role1 or Role2.

  ```sql
  GRANT SET USER (ANY WITH ROLES Role1, Role2) TO Steve, Joe
  ```

**Usage**

If no clause is specified, **ANY** is used by default.

If no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

If regranting the SET USER system privilege to a user, the effect of the regrant is cumulative.

By default, the SET USER system privilege is granted to the SYS_AUTH_SSO_ROLE compatibility role with the WITH NO ADMIN OPTION clause, if they exist.

The granting of the SET USER system privilege to a user only grants the potential to impersonate another user. Validation of the **at-least** criteria required to successfully impersonate another user does no occur until the **SETUSER** statement is issued.

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.
Permissions

- Requires the SET USER system privilege granted with administrative rights.
- Each target user specified (target_users_list) is 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.

GRANT System Privilege Statement

Grants specific system privileges to users, with or without administrative rights.

Syntax

```
GRANT system_privilege_name [, ...]
   TO userID [, ...]
   [ {WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Parameters

- `system_privilege` – must be the name of an existing system privilege.
- `userID` – must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate multiple userIDs with commas.
- `WITH NO ADMIN OPTION` – the user can manage the system privilege, but cannot grant the system privilege to another user.
- `WITH ADMIN ONLY OPTION` – If the WITH ADMIN ONLY OPTION clause is used, each `userID` is granted administrative privileges over each `system_privilege`, but NOT the `system_privilege` itself.
- `WITH ADMIN OPTION` – each `userID` is granted administrative privileges over each `system_privilege` in addition to all underlying system privileges of `system_privilege`.

Examples

- **Example 1** – grants the DROP CONNECTION system privilege to Joe with administrative privileges.
  
  ```sql
  GRANT DROP CONNECTION TO Joe WITH ADMIN OPTION
  ```

- **Example 2** – grants the CHECKPOINT system privilege to Sally with no administrative privileges.
  
  ```sql
  GRANT CHECKPOINT TO Sally WITH NO ADMIN OPTION
  ```

- **Example 3** – grants the MONITOR system privilege to Jane with administrative privileges only.
  
  ```sql
  GRANT MONITOR TO Jane WITH ADMIN ONLY OPTION
  ```
Usage
By default, if no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

Standards
- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions
Requires administrative privilege over the system privilege being granted.

List of All 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:

- ACCESS SERVER LS system privilege
- ALTER ANY INDEX system privilege
- ALTER ANY MATERIALIZED VIEW system privilege
- ALTER ANY OBJECT system privilege
- ALTER ANY OBJECT OWNER system privilege
- ALTER ANY PROCEDURE system privilege
- ALTER ANY SEQUENCE system privilege
- ALTER ANY TABLE system privilege
- ALTER ANY TEXT CONFIGURATION system privilege
- ALTER ANY TRIGGER system privilege
- ALTER ANY VIEW system privilege
- ALTER DATABASE system privilege
- ALTER DATATYPE system privilege
- BACKUP DATABASE system privilege
- CHANGE PASSWORD system privilege
- CHECKPOINT system privilege
- COMMENT ANY OBJECT system privilege
- CREATE ANY INDEX system privilege
- CREATE ANY MATERIALIZED VIEW system privilege
- CREATE ANY OBJECT system privilege
- CREATE ANY PROCEDURE system privilege
- CREATE ANY SEQUENCE system privilege
- CREATE ANY TABLE system privilege
• CREATE ANY TEXT CONFIGURATION system privilege
• CREATE ANY TRIGGER system privilege
• CREATE ANY VIEW system privilege
• CREATE DATATYPE system privilege
• CREATE EXTERNAL REFERENCE system privilege
• CREATE MATERIALIZED VIEW system privilege
• CREATE MESSAGE system privilege
• CREATE PROCEDURE system privilege
• CREATE PROXY TABLE system privilege
• CREATE TABLE system privilege
• CREATE TEXT CONFIGURATION system privilege
• CREATE VIEW system privilege
• DEBUG ANY PROCEDURE system privilege
• DELETE ANY TABLE system privilege
• DROP ANY INDEX system privilege
• DROP ANY MATERIALIZED VIEW system privilege
• DROP ANY OBJECT system privilege
• DROP ANY OBJECT PRIVILEGES system privilege
• DROP ANY PROCEDURE system privilege
• DROP ANY SEQUENCE system privilege
• DROP ANY TABLE system privilege
• DROP ANY TEXT CONFIGURATION system privilege
• DROP ANY VIEW system privilege
• DROP CONNECTION system privilege
• DROP DATATYPE system privilege
• DROP MESSAGE system privilege
• EXECUTE ANY PROCEDURE system privilege
• LOAD ANY TABLE system privilege
• INSERT ANY TABLE system privilege
• MANAGE ANY DBSPACE system privilege
• MANAGE ANY EVENT system privilege
• MANAGE ANY EXTERNAL ENVIRONMENT system privilege
• MANAGE ANY EXTERNAL OBJECT system privilege
• MANAGE ANY LDAP SERVER system privilege
• MANAGE ANY LOGIN POLICY system privilege
• MANAGE ANY MIRROR SERVER system privilege
• MANAGE ANY OBJECT PRIVILEGES system privilege
• MANAGE ANY SPATIAL OBJECT system privilege
• MANAGE ANY STATISTICS system privilege
• MANAGE ANY USER system privilege
• MANAGE ANY WEB SERVICE system privilege
• MANAGE AUDITING system privilege
• MANAGE MULTIPLEX system privilege
• MANAGE PROFILING system privilege
• MANAGE REPLICATION system privilege
• MANAGE ROLES system privilege
• MONITOR system privilege
• READ CLIENT FILE system privilege
• READ FILE system privilege
• REORGANIZE ANY OBJECT system privilege
• SELECT ANY TABLE system privilege
• SERVER OPERATOR system privilege
• SET ANY PUBLIC OPTION system privilege
• SET ANY SECURITY OPTION system privilege
• SET ANY SYSTEM OPTION system privilege
• SET ANY USER DEFINED OPTION system privilege
• SET USER system privilege (granted with ADMIN ONLY clause)
• TRUNCATE ANY TABLE system privilege
• UPDATE ANY TABLE system privilege
• UPGRADE ROLE system privilege
• USE ANY SEQUENCE system privilege
• VALIDATE ANY OBJECT system privilege
• WRITE CLIENT FILE system privilege
• WRITE FILE system privilege

**GRANT USAGE ON SEQUENCE Statement**

Grants USAGE privilege on a specified sequence.

**Syntax**

```
GRANT USAGE ON SEQUENCE sequence-name
TO userID [, ...]
```

**Parameters**

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

**Standards**

- **SQL** – syntax is a Persistent Stored Module feature.
• Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions
Requires one of:

• MANAGE ANY OBJECT PRIVILEGE system privilege.
• You own the sequence.

REVOKE CHANGE PASSWORD Statement
Removes the ability of a user to manage passwords and administer the system privilege.

You can revoke the CHANGE PASSWORD system privilege from any combination of users and roles granted.

Syntax
REVOKE [ ADMIN OPTION FOR ] CHANGE PASSWORD [(target_user_list | ANY | ANY WITH ROLES target_role_list )]
FROM userID [, ...]

Parameters
• target_user_list – if specified, must consist of existing users with login passwords and is the potential list of target users who can no longer have passwords managed by grantee users. Separate the user IDs in the list with commas.

• ANY – if specified, the potential list of target users for each grantee consists of all database users with login passwords.

• ANY WITH ROLES target_role_list – if specified, the target_role_list must consist of existing roles, and the potential list of target users for each grantee must consist of database users with login passwords that have a subset of roles in target_role_list. Separate the list of roles with commas.

• userID – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples
• Example 1 – removes the ability of Joe to manage the passwords of Sally or Bob.
  REVOKE CHANGE PASSWORD (Sally, Bob) FROM Joe

• Example 2 – if the CHANGE PASSWORD system privilege was originally granted to Sam with the WITH ADMIN OPTION clause, this example removes the ability of Sam to grant the CHANGE PASSWORD system privilege to another user, but still allows Sam to manage passwords for those users specified in the original GRANT CHANGE PASSWORD statement. However, if the CHANGE PASSWORD system privilege was originally
granted to Sam with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from Sam.

```
REVOKE ADMIN OPTION FOR CHANGE PASSWORD FROM Sam
```

**Usage**

Depending on how the CHANGE PASSWORD system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking the CHANGE PASSWORD system privilege has different results. If the CHANGE PASSWORD system privilege was originally granted with the WITH ADMIN OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement revokes only the ability to administer the CHANGE PASSWORD system privilege (that is, grant the system privilege to another user). The ability to actually manage passwords for other users remains. However, if the CHANGE PASSWORD system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement is semantically equivalent to revoking the entire CHANGE PASSWORD system privilege. Finally, if the CHANGE PASSWORD system privilege was originally granted with the WITH NO ADMIN OPTION clause, and the ADMIN OPTION FOR clause is included in the revoke statement, nothing is revoked because there were no administrative rights granted in the first place.

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.

**Permissions**

Requires the CHANGE PASSWORD system privilege granted with administrative rights.

**REVOKE CONNECT Statement**

Removes a user from the database.

**Syntax**

```
REVOKE CONNECT FROM userID [, ...]
```

**Parameters**

- `userID` – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

**Usage**

- Use system procedures or CREATE USER and DROP USER statements, not GRANT and REVOKE statements, to add and remove user IDs.
You cannot revoke the connect privileges from a user if he or she owns database objects, such as tables. Attempting to do so with a `REVOKE` statement, or `sp_droplogin` or `sp_iqdroplogin` stored procedure returns an error such as *Cannot drop a user that owns tables in runtime system.*

**Standards**

ANSI SQL – compliance level: Transact-SQL extension.

**Permissions**

Requires the MANAGE ANY USER system privilege.

**Note:** If revoking `CONNECT` permissions or revoking table permissions from another user, the target user cannot be connected to the database.

### REVOKE CREATE Statement

Removes CREATE permissions on the specified dbspace from the specified user IDs.

**Syntax**

```
REVOKE CREATE ON dbspace-name
FROM userID [, ...]
```

**Parameters**

- `userID` – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

**Examples**

- **Example 1** – revokes the CREATE privilege on dbspace `DspHist` from user `Smith`.
  ```
  REVOKE CREATE ON DspHist FROM Smith
  ```

- **Example 2** – revokes the CREATE permission on dbspace `DspHist` from user ID `fionat` from the database.
  ```
  REVOKE CREATE ON DspHist FROM fionat
  ```

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.

**Permissions**

Requires the MANAGE ANY DBSPACE system privilege.
REVOKE EXECUTE Statement

Removes EXECUTE permissions that were given using the GRANT statement.

Syntax

REVOKE EXECUTE ON [ owner.]procedure-name
FROM userID [, ...]

Parameters

- userID — must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL—Syntax is a Persistent Stored Module feature.
- Sybase—Syntax is supported by Adaptive Server Enterprise. User management and security models are different for v and SAP Sybase IQ.

Permissions

You either:

- Own the procedure, or
- Have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.

REVOKE Object-Level Privilege Statement

Removes object-level privileges that were given using the GRANT statement.

Syntax

REVOKE {permission [, ...]}
ON [ owner.]table-name
FROM userID [, ...]

permission

ALL [ PRIVILEGES ]
| ALTER
| DELETE
| INSERT
| REFERENCE [ ( column-name [, ...] ) ]
| SELECT [ ( column-name [, ...] ) ]
| UPDATE [ ( column-name, ... ) ]
| LOAD
| TRUNCATE
Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – prevents user Dave from inserting into the Employees table.
  
  ```sql
  REVOKE INSERT ON Employees FROM Dave
  ```

- **Example 2** – prevents user Dave from updating the Employees table.
  
  ```sql
  REVOKE UPDATE ON Employees FROM Dave
  ```

Standards

- SQL–Syntax is an entry-level feature.
- Sybase–Syntax is supported in Adaptive Server Enterprise.

Permissions

You either:

- Own the table, or
- Have the MANAGE ANY OBJECT PRIVILEGE system privilege granted with the GRANT OPTION clause.

REVOKE ROLE Statement

Removes a users membership in a role or his or her ability to administer the role.

Syntax

```
REVOKE [ADMIN OPTION FOR] ROLE role_name [,..]
    FROM grantee [,..]
```

- `role_name`:
  - `dbo`†††
  - `diagnostics`†††
  - `PUBLIC`†††
  - `rs_systabgroup`†††
  - `SA_DEBUG`†††
  - `SYS`†††
  - `SYS_AUTH_SA_ROLE`
  - `SYS_AUTH_SSO_ROLE`
  - `SYS_AUTH_DBA_ROLE`
  - `SYS_AUTH_RESOURCE_ROLE`
  - `SYS_AUTH_BACKUP_ROLE`
  - `SYS_AUTH_VALIDATE_ROLE`
  - `SYS_AUTH_WRITEFILE_ROLE`
  - `SYS_AUTH_WRITEFILECLIENT_ROLE`
  - `SYS_AUTH_READFILE_ROLE`
  - `SYS_AUTH_READFILECLIENT_ROLE`
| SYS_AUTH_PROFILE_ROLE  |
| SYS_AUTH_USER_ADMIN_ROLE   |
| SYS_AUTH_SPACE_ADMIN_ROLE  |
| SYS_AUTH_MULTIPLEX_ADMIN_ROLE |
| SYS_AUTH_OPERATOR_ROLE    |
| SYS_AUTH_PERMS_ADMIN_ROLE  |
| SYS_REPLICATE_ADMIN_ROLE††† |
| SYS_RUN_REPLICATE_ROLE†††  |
| SYS_SPATIAL_ADMIN_ROLE†††  |
| <user-defined role name>   |

†††The ADMIN OPTION FOR clause is not supported for system roles.

**Parameters**

- **role_name** – must already exist in the database. Separate multiple role names with commas.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **ADMIN OPTION FOR** – each userID must have been granted administrative privilege over the specified role_name.

**Note:** This clause revokes administrative privileges of the role only, not membership in the role, unless the role was originally granted with the WITH ADMIN ONLY OPTION clause. For roles granted with the WITH ADMIN ONLY OPTION clause, the ADMIN OPTION FOR clause is optional as it is semantically equivalent to revoking membership in a role in its entirety.

**Examples**

- **Example 1** – revokes the user-defined (standalone) role Role1 from User1.

  REVOKE ROLE Role1 FROM User1

  After you execute this command, User1 no longer has the rights to perform any authorized tasks using any system privileges granted to Role1.

- **Example 2** – revokes the ability for User1 to administer the compatibility role SYS_AUTH_WRITEFILE_ROLE.

  REVOKE ADMIN OPTION FOR ROLE SYS_AUTH_WRITEFILE_ROLE FROM User1

  User1 retains the ability to perform any authorized tasks granted by SYS_AUTH_WRITEFILE_ROLE.

**Standards**

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.
Permissions

Requires the MANAGE ROLES system privilege to revoke these roles:

- diagnostics
- dbo
- PUBLIC
- rs_systabgroup
- SA_DEBUG
- SYS
- SYS_RUN_REPLICATE_ROLE
- SYS_SPATIAL_ADMIN_ROLE

Requires administrative privilege over the role to revoke these roles:

- SYS_AUTH_SA_ROLE
- SYS_AUTH_SSO_ROLE
- SYS_AUTH_DBA_ROLE
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE
- SYS_AUTH_WRITEFILE_ROLE
- SYS_AUTH_WRITEFILECLIENT_ROLE
- SYS_AUTH_READFILE_ROLE
- SYS_AUTH_READFILECLIENT_ROLE
- SYS_AUTH_PROFILE_ROLE
- SYS_AUTH_USER_ADMIN_ROLE
- SYS_AUTH_SPACE_ADMIN_ROLE
- SYS_AUTH_MULTIPLEX_ADMIN_ROLE
- SYS_AUTH_OPERATOR_ROLE
- SYS_AUTH_PERMS_ADMIN_ROLE
- <user-defined role name>

REVOKE SET USER Statement

Removes the ability for one user to impersonate another user and to administer the SET USER system privilege.

Syntax

REVOKE [ ADMIN OPTION FOR ] SETUSER
(target_user_list | ANY | ANY WITH ROLES target_role_list ])
FROM userID [, ...]
Parameters

- **target_user_list** – must consist of existing users with login passwords and is the potential list of target users who can no longer be impersonated by grantee users. Separate the user IDs in the list with commas.
- **ANY** – If specified, the potential list of target users for each grantee consists of all database users with login passwords.
- **ANY WITH ROLES target_role_list** – If specified, the target_role_list must consist of existing roles, and the potential list of target users for each grantee must consist of database users with login passwords that have a subset of roles in target_role_list. Separate the list of roles with commas.
- **userID** – Each userID must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.

Examples

- **Example 1** – stops Bob from being able to impersonate Sally or Bob.
  
  ```sql
  REVOKE SET USER (Sally, Bob) FROM Bob
  ```

- **Example 2** – if the SET USER system privilege was originally granted to Sam with the WITH ADMIN OPTION clause, this example removes the ability of Sam to grant the SET USER system privilege to another user, but still allows Sam to impersonate those users already granted to him or her. However, if the SET USER system privilege was originally granted to Sam with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from Sam.
  
  ```sql
  REVOKE ADMIN OPTION FOR SET USER FROM Sam
  ```

Usage

Depending on how the SET USER system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking the SET USER system privilege has different results. If you the SET USER system privilege was originally granted with the WITH ADMIN OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement revokes only the ability to administer the SET USER system privilege (that is, grant the system privilege to another user). The ability to actually impersonate another user remains. However, if the SET USER system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement is semantically equivalent to revoking the entire SET USER system privilege. Finally, if the SET USER system privilege was originally grant with the WITH NO ADMIN OPTION clause, and the ADMIN OPTION FOR clause is included in the revoke statement, nothing is revoked because there were no administrative system privileges granted in the first place.
Standards
ANSI SQL – Compliance level: Transact-SQL extension.

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

REVOKE System Privilege Statement
Removes specific system privileges from specific users and the right to administer the privilege.

Syntax
REVOKE [ADMIN OPTION FOR] system_privilege [,..] FROM userID [,..]

Parameters
• system_privilege – must be an existing system privilege.
• userID – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
• ADMIN OPTION FOR – each system_privilege must currently be granted to each userID specified with administrative privileges.

Note: This clause revokes only the administrative privileges of the system privilege; the system privilege itself remains granted. However, if the system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, the ADMIN OPTION FOR clause completely revokes the system privilege. Under this scenario, use of the ADMIN OPTION FOR clause is not required to revoke administrative privileges.

Examples
• Example 1 – revokes the BACKUP DATABASE system privilege from user Jim.
  REVOKE BACKUP DATABASE FROM Jim

• Example 2 – assuming the BACKUP DATABASE system privilege was originally granted to user Jim with the WITH ADMIN OPTION clause, this example revokes the ability to administer the BACKUP DATABASE system privilege from user Jim. The ability to perform tasks authorized by the system privilege remains. However, if the BACKUP DATABASE system privilege was originally granted to user Jim with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from user Jim.
  REVOKE ADMIN OPTION FOR BACKUP DATABASE FROM Jim
Usage
Depending on how the system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking a system privilege has different results. If you the system privilege was originally granted with the WITH ADMIN OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement revokes only the ability to administer the system privilege (that is, grant the system privilege to another user). The ability to actually use the system privilege remains. However, if the system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement is semantically equivalent to revoking the entire system privilege. Finally, if the system privilege was originally grant with the WITH NO ADMIN OPTION clause, and the ADMIN OPTION FOR clause is included in the revoke statement, nothing is revoked because there were no administrative system privileges granted in the first place.

Standards
- SQL – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – syntax is not supported by Adaptive Server Enterprise.

Permissions
Requires administrative privilege over the system privilege being revoked.

List of All 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:

- ACCESS SERVER LS system privilege
- ALTER ANY INDEX system privilege
- ALTER ANY MATERIALIZED VIEW system privilege
- ALTER ANY OBJECT system privilege
- ALTER ANY OBJECT OWNER system privilege
- ALTER ANY PROCEDURE system privilege
- ALTER ANY SEQUENCE system privilege
- ALTER ANY TABLE system privilege
- ALTER ANY TEXT CONFIGURATION system privilege
- ALTER ANY TRIGGER system privilege
- ALTER ANY VIEW system privilege
- ALTER DATABASE system privilege
- ALTER DATATYPE system privilege
- BACKUP DATABASE system privilege
- CHANGE PASSWORD system privilege
- CHECKPOINT system privilege
- COMMENT ANY OBJECT system privilege
- CREATE ANY INDEX system privilege
- CREATE ANY MATERIALIZED VIEW system privilege
- CREATE ANY OBJECT system privilege
- CREATE ANY PROCEDURE system privilege
- CREATE ANY SEQUENCE system privilege
- CREATE ANY TABLE system privilege
- CREATE ANY TEXT CONFIGURATION system privilege
- CREATE ANY TRIGGER system privilege
- CREATE ANY VIEW system privilege
- CREATE DATATYPE system privilege
- CREATE EXTERNAL REFERENCE system privilege
- CREATE MATERIALIZED VIEW system privilege
- CREATE MESSAGE system privilege
- CREATE PROCEDURE system privilege
- CREATE PROXY TABLE system privilege
- CREATE TABLE system privilege
- CREATE TEXT CONFIGURATION system privilege
- CREATE VIEW system privilege
- DEBUG ANY PROCEDURE system privilege
- DELETE ANY TABLE system privilege
- DROP ANY INDEX system privilege
- DROP ANY MATERIALIZED VIEW system privilege
- DROP ANY OBJECT system privilege
- DROP ANY PROCEDURE system privilege
- DROP ANY SEQUENCE system privilege
- DROP ANY TABLE system privilege
- DROP ANY TEXT CONFIGURATION system privilege
- DROP ANY VIEW system privilege
- DROP CONNECTION system privilege
- DROP DATATYPE system privilege
- DROP MESSAGE system privilege
- EXECUTE ANY PROCEDURE system privilege
- LOAD ANY TABLE system privilege
- INSERT ANY TABLE system privilege
- MANAGE ANY DBSPACE system privilege
- MANAGE ANY EVENT system privilege
- MANAGE ANY EXTERNAL ENVIRONMENT system privilege
• MANAGE ANY EXTERNAL OBJECT system privilege
• MANAGE ANY LDAP SERVER system privilege
• MANAGE ANY LOGIN POLICY system privilege
• MANAGE ANY MIRROR SERVER system privilege
• MANAGE ANY OBJECT PRIVILEGES system privilege
• MANAGE ANY SPATIAL OBJECT system privilege
• MANAGE ANY STATISTICS system privilege
• MANAGE ANY USER system privilege
• MANAGE ANY WEB SERVICE system privilege
• MANAGE AUDITING system privilege
• MANAGE MULTIPLEX system privilege
• MANAGE PROFILING system privilege
• MANAGE REPLICATION system privilege
• MANAGE ROLES system privilege
• MONITOR system privilege
• READ CLIENT FILE system privilege
• READ FILE system privilege
• REORGANIZE ANY OBJECT system privilege
• SELECT ANY TABLE system privilege
• SERVER OPERATOR system privilege
• SET ANY PUBLIC OPTION system privilege
• SET ANY SECURITY OPTION system privilege
• SET ANY SYSTEM OPTION system privilege
• SET ANY USER DEFINED OPTION system privilege
• SET USER system privilege (granted with ADMIN ONLY clause)
• TRUNCATE ANY TABLE system privilege
• UPDATE ANY TABLE system privilege
• UPGRADE ROLE system privilege
• USE ANY SEQUENCE system privilege
• VALIDATE ANY OBJECT system privilege
• WRITE CLIENT FILE system privilege
• WRITE FILE system privilege

REVOKE USAGE ON SEQUENCE Statement
Removes USAGE privilege on a specified sequence.

Syntax
REVOKE USAGE ON SEQUENCE sequence-name
FROM userID [,...]

Appendix: SQL Reference
Parameters

• **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

• SQL – syntax is a Persistent Stored Module feature.
• Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires one of:

• MANAGE ANY OBJECT PRIVILEGE system privilege.
• You own the sequence.

**SET OPTION Statement**

Changes database options.

**Syntax**

```
SET [ EXISTING ] [ TEMPORARY ] OPTION ...
[ userid. | PUBLIC.]option-name = [ option-value ]
```

Parameters

• **userid** – identifier, string, or host-variable
• **option-name** – identifier, string, or host-variable
• **option-value** – host-variable (indicator allowed), string, identifier, or number

Examples

• **Example 1** – Set the DATE_FORMAT option:
  ```sql
  SET OPTION public.date_format = 'Mmm dd yyyy'
  ```

• **Example 2** – Set the WAIT_FOR_COMMIT option to on:
  ```sql
  SET OPTION wait_for_commit = 'on'
  ```

• **Example 3** – Embedded SQL examples:
  ```sql
  EXEC SQL SET OPTION :user.:option_name = :value;
  EXEC SQL SET TEMPORARY OPTION Date_format = 'mm/dd/yyyy';
  ```
Usage

The SET OPTION statement is used to change options that affect the behavior of the database and its compatibility with Transact-SQL. Setting the value of an option can change the behavior for all users or an individual user, in either a temporary or permanent scope.

The classes of options are:

- General database options
- Transact-SQL compatibility database options

Specifying either a user ID or the PUBLIC user ID determines whether the option is set for an individual user, a role represented by userid, or the PUBLIC user ID (the role to which all users are a member). If the option applies to a role ID, option settings are not inherited by members of the role—the change is applied only to the role ID. If no role is specified, the option change is applied to the currently logged-in user ID that issued the SET OPTION statement.

For example, this statement applies an option change to the PUBLIC user ID:

```
SET OPTION Public.login_mode = standard
```

Only users with the SET ANY SYSTEM OPTION system privilege have the ability to set a SYSTEM option for the PUBLIC user ID. Only users with the SET ANY SECURITY OPTION system privilege have the ability to set a SECURITY option for the PUBLIC user ID.

In Embedded SQL, only database options can be set temporarily.

Changing the value of an option for the PUBLIC user ID sets the value of the option for any user that has not set its own value. Option values cannot be set for an individual user ID unless there is already a PUBLIC user ID setting for that option.

Users cannot set the options of another user, unless they have the SET ANY PUBLIC OPTION system privilege.

Users can use the SET OPTION statement to change the values for their own user IDs. Setting the value of an option for a user ID other than your own is permitted only if you have the SET ANY PUBLIC OPTION system privilege.

If you use the EXISTING keyword, option values cannot be set for an individual user ID unless there is already a PUBLIC user ID setting for that option.

Adding the TEMPORARY keyword to the SET OPTION statement changes the duration that the change takes effect. Without the TEMPORARY keyword, an option change is permanent: it does not change until it is explicitly changed using SET OPTION.

When SET TEMPORARY OPTION is applied using an individual user ID, the new option value is in effect as long as that user is logged in to the database.

When SET TEMPORARY OPTION is used with the PUBLIC user ID, the change is in place for as long as the database is running. When the database is shut down, TEMPORARY options for the PUBLIC user ID revert back to their permanent value.
Temporarily setting an option for the PUBLIC user ID, as opposed to setting the value of the option permanently, offers a security advantage. For example, when the \texttt{LOGIN\_MODE} option is enabled, the database relies on the login security of the system on which it is running. Enabling the option temporarily means a database relying on the security of a Windows domain is not compromised if the database is shut down and copied to a local machine. In that case, the temporary enabling of \texttt{LOGIN\_MODE} reverts to its permanent value, which might be Standard, a mode in which integrated logins are not permitted.

If \textit{option-value} is omitted, the specified option setting is deleted from the database. If it was a personal option setting, the value used reverts to the PUBLIC setting. If a \texttt{TEMPORARY} option is deleted, the option setting reverts to the permanent setting.

\textbf{Note:} For all database options that accept integer values, SAP Sybase IQ truncates any decimal \textit{option-value} setting to an integer value. For example, the value 3.8 is truncated to 3.

The maximum length of \textit{option-value} when set to a string is 127 bytes.

\textbf{Warning!} Changing option settings while fetching rows from a cursor is not supported, as it can lead to unpredictable behavior. For example, changing the \texttt{DATE\_FORMAT} setting while fetching from a cursor returns different date formats among the rows in the result set. Do not change option settings while fetching rows.

\section*{Standards}

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. SAP Sybase IQ does support some Adaptive Server Enterprise options using the \texttt{SET} statement.

\section*{Permissions}

No specific system privileges are required to set your own options. However, the \texttt{SET ANY SYSTEM OPTION} system privilege is required to set system database options for PUBLIC. The \texttt{SET ANY SECURITY OPTION} system privilege is required to set security database options for PUBLIC. Finally, the \texttt{SET ANY PUBLIC OPTION} system privilege is required to set database options for another user.

\section*{SETUSER Statement}

Allows a user to temporarily assume the roles and system privileges of another user (also known as impersonation) to perform operations, provided they already have the minimum required privileges to perform the task to begin with.

\textbf{Note:} The \texttt{SET USER} system privilege is two words; the \texttt{SETUSER} statement is one word.

\section*{Syntax}

\begin{verbatim}
SETUSER userID
\end{verbatim}
Usage

At least criteria validation occurs when the SETUSER statement is executed, not when the SET USER system privilege is granted.

UserID must be the name of an existing user or role that has a login password.

To terminate a successful impersonation, issue the SETUSER statement without specifying a userID.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

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 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. For example, User1 is granted Role1 with the WITH ADMIN OPTION clause, User2 is granted Role1 with the WITH ADMIN ONLY clause, and User3 is granted Role1 with the WITH NO ADMIN OPTION clause. User1 and User2 are said to be granted Role1 with similar administrative rights. User1 and User2 are also said to be granted Role1 with higher administrative rights that User3.

4. If the target user has been granted a system privilege which supports extensions, the clauses used to grant the system privilege to the impersonator are a super-set of those used for the target user. Currently, only the SET USER and CHANGE PASSWORD system privileges support extensions.

Note:

• 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, Role1, Role2, respectively, while the target list grants of the target user contain User1,
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 themselves. Therefore, if the target user has been 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.

VALIDATE LDAP SERVER Statement

Validates changes to the settings of existing LDAP server configuration objects before applying them.

This statement is useful for an administrator when setting up a new server to use LDAP user authentication, and for diagnosing problems between the LDAP server configuration object and the external LDAP server. Any connection made by the VALIDATE LDAP SERVER statement is temporary and is closed by the end of the statement.

Syntax

```
VALIDATE LDAP SERVER [ ldapua-server-name | ldapua-server-attribs ]
[ CHECK userid [ user-dn-string ] ]
```

```
ldapua-server-attribs:
SEARCH DN
  URL { 'URL_string' | NULL }
  | ACCESS ACCOUNT { 'DN_string' | NULL }
  | IDENTIFIED BY ( 'password' | NULL )
  | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
| AUTHENTICATION URL { 'URL_string' | NULL }
| CONNECTION TIMEOUT timeout_value
| CONNECTION RETRIES retry_value
| TLS { ON | OFF }
```

Parameters

- `ldapua-server-name` – identifies the LDAP server configuration object.
- `URL` – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP
URL syntax before it is stored in the ISYSLDAPSERVER system table. The maximum size for this string is 1024 bytes.

- **ACCESS ACCOUNT** – a user created on the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect to the LDAP server. This user has permissions within the LDAP server to search for DNs by user ID in the locations specified by the SEARCH DN URL. The maximum size for this string is 1024 bytes.

- **IDENTIFIED BY** – provides the password associated with the ACCESS ACCOUNT user. The password is stored using symmetric encryption on disk. Use the value NULL to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.

- **IDENTIFIED BY ENCRYPTED** – configures the password associated with the ACCESS ACCOUNT distinguished name in an encrypted format. The binary value is the encrypted password and is stored on disk as is. Use the value NULL to clear the password and set it to none. The maximum size of the binary is 289 bytes.

- **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for <URL_string> and is validated for correct LDAP URL syntax before it is stored in ISYSLDAPSERVER system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.

- **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.

- **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase 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 for both DN searches and authentication. When set to ON, the TLS protocol is used and the URL begins with "ldap://". When set to OFF (or not specified), Secure LDAP protocol is used and the URL begins with “ldaps://”. When using the TLS protocol, specify the database security option TRUSTED_CERTIFICATES_FILE with a file name containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.

- **CHECK userID** – the userID whose existence is validated on the LDAP server.

- **user-dn-string** – compares a user's DN value with the user ID for verification purposes.

**Examples**

- **Example 1** – assume the apps_primary LDAP server configuration object was created as follows:

```sql
SET OPTION PUBLIC.login_mode = ‘Standard,LDAPUA’
CREATE LDAP SERVER apps_primary
```
This statement validates the existence of a userID `myusername` by using the optional CHECK clause to compare the userID to the expected user distinguished name (enclosed in quotation marks) on the `apps_primary` LDAP server configuration object.

```sql
VALIDATE LDAP SERVER apps_primary
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com'
```

**Usage**

When validating the LDAP server configuration object by name, definitions from prior `CREATE LDAP SERVER` and `ALTER LDAP SERVER` statements are used. Alternately, when `ldapua-server-attributes` are specified instead of the LDAP server configuration object name, the specified attributes are validated. When `ldapua-server-attributes` are specified, the URLs are parsed to identify syntax errors, and statement processing stops is a syntax error is detected.

Whether using an LDAP server configuration object name or a successfully parsed set of `ldapua-server-attributes`, a connection to the external LDAP server is attempted. If the parameter `ACCESS ACCOUNT` and a password are specified, the values are used to establish the connection to the `SEARCH DN` URL. This the `SEARCH DN` URL, `ACCESS ACCOUNT`, and `ACCESS ACCOUNT` password.

When using the optional CHECK clause, the userID is used in the search to validate the existence of the user on the external LDAP server. When the expected DN value for a given user is known, the value can be specified, and is compared with the result of the search to determine success or failure.

**Standards**

ANSI SQL – Compliance level: Transact-SQL extension.
Permissions
Requires the MANAGE ANY LDAP SERVER system privilege.

Database Options
Database options customize and modify database behavior.

LOGIN_MODE Option
Controls the use of integrated logins for the database.

Allowed Values
- Standard – the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- Mixed – both integrated logins and standard logins are allowed.
- Integrated – all logins to the database must be made using integrated logins.
- Kerberos – all logins to the database must be made using Kerberos logins.
- LDPAUA – all logins to the database must be made using LDAP logins.

Note: Mixed is equivalent to "Standard,Integrated".

Default
Standard

Scope
Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description
Values are case-insensitive:

Warning!
- Restricting the LOGIN_MODE to a single mode in a mixed environment (for example, Integrated only or LDPAUA only) restricts connections to only those users who have been granted the corresponding login mapping. Attempting to connect using other methods generates an error. The only exceptions to this are users with full administrative rights (SYS_AUTH_DBA_ROLE or SYS_AUTH_SSO_ROLE).
- Restricting the LOGIN_MODE to LDPAUA only may result in a configuration where no users can connect to the server if no user or login policy exists that permits LDPAUA. Use the command line switch -al-user-id-list with the start_iq utility to recover from this situation.
MIN_ROLEAdmins Option

Configures the minimum number of required administrators for all roles.

Allowed Values
1 – 10

Default
1

Scope
Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description
This option sets the minimum number of required administrators for all roles. This value applies to the minimum number of role administrators for each role, not the minimum number or role administrators for the total number of roles. When dropping roles or users, this value ensures that you never create a scenario where there are no users and roles left with sufficient system privilege to manage the remaining users and roles.

TRUSTED_CERTIFICATES_FILE Option

Specifies the trust relationship for outbound Transport Layer Security (TLS) connections made by LDAP User Authentication, INC, and MIPC connections.

Allowed Values
A valid network path to the location of a TXT file containing the list of trusted certificate authorities that sign server certificates.

Default
NULL, meaning that no outbound TLS connection can be started because there are no trusted certificate authorities.

Scope
Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.
**Description**
This option identifies the path to the location of the list of trusted certificate authorities. The list must be stored in a TXT file. The file may be shared in a location in a Windows environment on the local drive to be used by all SAP Sybase applications on that machine.

-al iqsrv16 Server Option
Extends LOGIN_MODE for LDAPUA only to a select number of users using Standard authentication

**Syntax**
-al "user1;user2;user3" server_name.cfg database-name.db

**Remarks**
- Up to five user IDs can be specified, separated by semi-colons, and enclosed in double quotation marks.
- When run at the server level, the -al switch remains in effect until the next time the server is restarted.

-al iqsrv16 Database Option
Extends LOGIN_MODE for LDAPUA only to a select number of users using Standard authentication

**Syntax**
-al "user1;user2;user3" server_name.cfg database_name.db

**Remarks**
- Up to five user IDs can be specified, separated by semi-colons, and enclosed in double quotation marks.
- When run at the database level, it remains in effect until the next time the database is stopped/started.

VERIFY_PASSWORD_FUNCTION Option
Specifies a user-supplied authentication function that can be used to implement password rules.

**Allowed Values**
String

**Default**
" (the empty string). (No function is called when a password is set.)
Scope
Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description
When the VERIFY_PASSWORD_FUNCTION option value is set to a valid string, the statement GRANT CONNECT TO userid IDENTIFIED BY password calls the function specified by the option value.

The option value requires the form owner.function_name to prevent users from overriding the function.

The function takes two parameters:
- user_name VARCHAR(128)
- new_pwd VARCHAR(255)

The return value type is VARCHAR(255).

If VERIFY_PASSWORD_FUNCTION is set, you cannot specify more than one userid and password with the GRANT CONNECT statement.

Example
The following sample code defines a table and a function and sets some login policy options. Together they implement advanced password rules that include requiring certain types of characters in the password, disallowing password reuse, and expiring passwords. The function is called by the database server with the verify_password_function option when a user ID is created or a password is changed. The application can call the procedure specified by the post_login_procedure option to report that the password should be changed before it expires.

```sql
-- only DBA should have privileges on this table
CREATE TABLE DBA.t_pwd_history(
    pk          INT         DEFAULT AUTOINCREMENT PRIMARY KEY,
    user_name   CHAR(128),  -- the user whose password is set
    pwd_hash    CHAR(32)    ; -- hash of password value to detect
                        -- duplicate passwords

-- called whenever a non-NULL password is set
-- to verify the password conforms to password rules
CREATE FUNCTION DBA.f_verify_pwd( uid     VARCHAR(128),
                                    new_pwd VARCHAR(255) )
RETURNS VARCHAR(255)
BEGIN
    -- enforce password rules
```

SAP Sybase IQ
-- enforce minimum length (can also be done with
-- min_password_length option)
IF length( new_pwd ) < 6 THEN
    RETURN 'password must be at least 6 characters long';
END IF;

-- number of lowercase characters IN new_pwd
SELECT count(*) INTO num_lower_chars
FROM pwd_chars WHERE CAST( c AS BINARY ) BETWEEN 'a' AND 'z';

-- enforce rules based on characters contained in new_pwd
IF ( SELECT count(*) FROM pwd_chars WHERE c BETWEEN '0' AND '9' ) < 1 THEN
    RETURN 'password must contain at least one numeric digit';
ELSEIF length( pwd_alpha_only ) < 2 THEN
    RETURN 'password must contain at least two letters';
ELSEIF num_lower_chars = 0
    OR length( pwd_alpha_only ) - num_lower_chars = 0 THEN
    RETURN 'password must contain both upper- and lowercase
characters';
END IF;

-- not the same as any user name
-- (this could be modified to check against a disallowed words
table)
IF EXISTS( SELECT * FROM SYS.SYSUSER
    WHERE lower( user_name ) IN
    ( lower( pwd_alpha_only ),
    lower( new_pwd ) ) ) THEN
    RETURN 'password or only alphabetic characters in password '
    ||
    'must not match any user name';
END IF;

-- not the same as any previous password for this user
IF EXISTS( SELECT * FROM t_pwd_history
    WHERE user_name = uid
    AND pwd_hash = hash( uid || new_pwd, 'md5' ) ) THEN
    RETURN 'previous passwords cannot be reused';
END IF;

-- save the new password
INSERT INTO t_pwd_history( user_name, pwd_hash )
VALUES( uid, hash( uid || new_pwd, 'md5' ) );

RETURN( NULL );
END;

ALTER FUNCTION DBA.f_verify_pwd SET HIDDEN;
GRANT EXECUTE ON DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';

-- All passwords expire in 180 days. Expired passwords can be changed
-- by the user using the NewPassword connection parameter.
ALTER LOGIN POLICY DEFAULT password_life_time = 180;
-- If an application calls the procedure specified by the post_login_procedure option, then the procedure can be used to warn the user that their password is about to expire. In particular,
-- Interactive SQL calls the post_login_procedure.
ALTER LOGIN POLICY DEFAULT password_grace_time = 30;

To turn the option off, set it to the empty string:

SET OPTION PUBLIC.VERIFY_PASSWORD_FUNCTION = ''

**MIN_PASSWORD_LENGTH Option**

Sets the minimum length for new passwords in the database.

*Allowed Values*

Integer greater than or equal to zero

The value is in bytes. For single-byte character sets, this is the same as the number of characters.

*Default*

3 characters

*Scope*

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

*Description*

This option imposes a minimum length on all new passwords for greater security. Existing passwords are not affected.

*Example*

Set the minimum length for new passwords to 6 bytes:

SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 6

**-gk iqsrv16 database server option**

Sets the privileges required to stop the database server.

*Syntax*

iqsrv16 -gk { DBA | all | none } ...
**Allowed values**

- **DBA** – Only users with the SERVER OPERATOR system privilege can stop the database server. This is the default for the network server.
- **all** – No privileges are required to shut down the database server.
- **none** – The database server cannot be stopped.

**Applies to**
All operating systems and database servers.

**Remarks**
The `-gd` database server option applies to the `dbstop` utility as well as to the following statements:

- ALTER DATABASE *dbname* FORCE START statement.
- STOP DATABASE statement

**-gl iqsrv16 Server Option**
Set the permission required to load data using **LOAD TABLE**.

**Syntax**

```
-gl level
```

**Remarks**
The **LOAD TABLE** statement reads files from the database server machine. To control access to the file system using these statements, the `-gl` command-line switch allows you to control the level of database permission that is required to use these statements. `level` is either:

- **DBA** – only users with the LOAD ANY TABLE, ALTER ANY TABLE or ALTER ANY OBJECT system privilege can load data.
- **ALL** – all users can load data.
- **NONE** – data cannot be loaded.

You can use either uppercase and lowercase syntax for the options.

The default settings are **all** for servers started with `start_iq` and **dba** for other servers. SAP Sybase recommends that, for consistency with earlier versions, you use the **all** value on all systems. The **all** setting is used in the `iqdemo.cfg` and `default.cfg` configuration files.
-gu iqsrv16 database server option

Sets the privilege required for executing database file administration statements such as for creating or dropping databases.

**Syntax**

\`
iqsrv16 -gu { all | none | DBA | utility_db } ...
``

**Allowed values**

<table>
<thead>
<tr>
<th>-gu option</th>
<th>Effect</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td>This option is deprecated. Anyone can execute file administration statements.</td>
<td>Any database including utility database</td>
</tr>
<tr>
<td>none</td>
<td>Executing file administration statements is not allowed.</td>
<td>Any database including utility database</td>
</tr>
<tr>
<td>DBA</td>
<td>Only users with the SERVER OPERATOR system privilege can execute file administration statements</td>
<td>Any database including utility database</td>
</tr>
<tr>
<td>utility_db</td>
<td>Only the users who can connect to the utility database can execute file administration statements</td>
<td>Only the utility database</td>
</tr>
</tbody>
</table>

**Default**

DBA

**Applies to**

All operating systems and database servers.

**Remarks**

Restricts the users who can execute the following database file administration statements:

- ALTER DATABASE dbfile ALTER TRANSACTION LOG
- CREATE DATABASE statement
- CREATE DECRYPTED DATABASE statement
- CREATE DECRYPTED FILE statement
- CREATE ENCRYPTED DATABASE statement
- CREATE ENCRYPTED FILE statement
- DROP DATABASE statement
• RESTORE DATABASE statement.

When utility_db is specified, these statements can only be run from the utility database. When DBA is specified, these statements can only be run by a user with the SERVER OPERATOR system privilege. When none is specified, no user can execute these statements.

**Examples**

To prevent the use of the file administration statements, start the database server using the none privilege level of the -gu option. The following command starts a database server and names it TestSrv. It loads the mytestdb.db database, but prevents anyone from using that server to create or delete a database, or execute any other file administration statement regardless of their resource creation rights, or whether they can load and connect to the utility database.

```
iqsrv16 -n TestSrv -gu none c:\mytestdb.db
```

To permit only the users knowing the utility database password to execute file administration statements, start the server by running the following command.

```
iqsrv16 -n TestSrv -su secret -gu utility_db
```

The following command starts Interactive SQL as a client application, connects to the server named TestSrv, loads the utility database, and connects the user.

```
dbIsql -c
"UID=DBA;PWD=secret;DBN=utility_db;Host=host1;Server=TestSrv"
```

Having executed the above command successfully, the user connects to the utility database, and can execute file administration statements.

**-sk iqsrv16 database server option**

Specifies a system secure feature key that can be used to allow access to features that are secured for the database server.

**Syntax**

```
iqsrv16 -sk key ...
```

**Applies to**

All operating systems and database servers.

**Remarks**

When you secure features for a database server by using the -sf option, you can also include the -sk option, which specifies a key that can be used with the sp_use_secure_feature_key system procedure to allow access to secured features for a connection. That connection can also use the sa_server_option system procedure to modify the features or feature sets that are secured for all databases running on the database server.

The key must be a non-empty string of at least six characters, and it cannot contain double quotes, control characters (any character less than 0x20), or backslashes. There is a limit of 1000 secure feature keys per database.
If the value for the authorization_key parameter of the sp_use_secure_feature_key system procedure is set to any value other than the one specified by -sk, no error is given and the features specified by -sf remain secured for the connection.

If you specify -sk without -sf, only the default secure features are enabled, but you can use the system secure feature key while the database server is running to change the secure feature settings.

**Example**

The following command starts a database server named secure_server with the backup feature secured. The key specified by the -sk option can be used later to allow access to these features for a specific connection.

```
iqsrv16 -n secure_server -sf backup -sk j978kls12
```

Setting the authorization_key parameter to the value specified by -sk for a connection to a database running on the secure_server database server allows that connection to perform backups or change the features that are secured on the secure_server database server:

```
CALL sp_use_secure_feature_key ( 'MyKey' , 'j978kls12' );
```

The user can then secure all features for databases running on secure_server by executing the following statement:

```
CALL sa_server_option( 'SecureFeatures', 'all' );
```

### -sf iqsrv16 database server option

Controls whether users have access to features for databases running on the current database server. A secured feature can only be accessed by a user with appropriate privileges, while an unsecured feature can be accessed by all users.

**Syntax**

```
iqsrv16 -sf feature-list ...
```

**feature-list :**

```
feature-name | feature-set [ ,feature-name | feature-set ] ...
```

<table>
<thead>
<tr>
<th>Feature set</th>
<th>Included features (feature sets in bold)</th>
</tr>
</thead>
<tbody>
<tr>
<td>none</td>
<td>All features are unsecured except manage_features, manage_keys, and disk_sandbox.</td>
</tr>
<tr>
<td>manage_server</td>
<td>processor_affinity</td>
</tr>
<tr>
<td>Feature set</td>
<td>Included features (feature sets in bold)</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>manage_security</td>
<td>manage_features</td>
</tr>
<tr>
<td></td>
<td>manage_keys</td>
</tr>
<tr>
<td></td>
<td>manage_disk_sandbox</td>
</tr>
<tr>
<td>server_security</td>
<td>disk_sandbox</td>
</tr>
<tr>
<td></td>
<td>trace_system_event</td>
</tr>
<tr>
<td>Feature set</td>
<td>Included features (feature sets in bold)</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>
| all         | client –
|             |   read_client_file
|             |   write_client_file
|             | remote –
|             |   remote_data_access
|             |   send_udp
|             |   send_email
|             |   web_service_client
|             | local –
|             |   • local_call –
|             |     cmdshell
|             |     external_procedure
|             |     java
|             |   • local_db –
|             |     backup
|             |     restore
|             |     database
|             |     dbspace
|             |   • local_env –
|             |     getenv
|             |   • local_io –
|             |     create_trace_file
|             |     read_file
|             |     write_file
|             |     directory
|             |     sp_list_directory
|             |     sp_create_directory
|             |     sp_copy_directory
|             |     sp_move_directory
|             |     sp_delete_directory
|             |     sp_copy_file
<table>
<thead>
<tr>
<th>Feature set</th>
<th>Included features (feature sets in bold)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sp_move_file</td>
</tr>
<tr>
<td></td>
<td>sp_delete_file</td>
</tr>
<tr>
<td></td>
<td>• local_log –</td>
</tr>
<tr>
<td></td>
<td>request_log</td>
</tr>
<tr>
<td></td>
<td>console_log</td>
</tr>
<tr>
<td></td>
<td>webclient_log</td>
</tr>
</tbody>
</table>

**Parameters**

- **none** – Specifies that no features are secured.
- **manage_server** – Prevents users from accessing all database server-related features. This set consists of the following features:
  - **processor_affinity** – Prevents users from changing the processor affinity (the number of logical processors being used) of the database server.
  - **manage_security** – Prevents users from accessing features that allow the management of database server security. By default, these features are secured.
  - **manage_features** – Prevents users from modifying the list of features that can be secured on the database server.
  - **manage_keys** – Prevents the creation, modification, deletion, or listing of secure feature keys.

A user that has access to the `manage_keys` feature but not the `manage_features` feature cannot define a key with more secure features than those assigned to the user.

- **manage_disk_sandbox** – Prevents users from temporarily changing disk sandbox settings by using the `sa_server_option` system procedure or the `sa_db_option` system procedure. The `manage_disk_sandbox` secure feature cannot be turned off for all databases or users—it can only be turned off for individual connections by using the `sp_use_secure_feature_key` system procedure.
- **server_security** – Prevents users from accessing features that can temporarily bypass security settings. By default, these features are secured.
  - **disk_sandbox** – Prevents users from performing read-write file operations on the database outside the directory where the main database file is located.
  - **trace_system_event** – Prevents users from creating user-defined trace events.
- **all** – Prevents users from accessing the following groups:
  - **client** – Prevents users from accessing all features that allow access to client-related input and output. This feature controls access to the client computing environment. This set consists of the following features:
• **read_client_file** – Prevents the use of statements that can cause a client file to be read. For example, the READ_CLIENT_FILE function and the LOAD TABLE statement.

• **write_client_file** – Prevents the use of all statements that can cause a client file to be written to. For example, the UNLOAD statement and the WRITE_CLIENT_FILE function.

• **remote** – Prevents users from accessing all features that allow remote access or communication with remote processes. This set consists of the following features:

  • **remote_data_access** – Prevents the use of any remote data access services, such as proxy tables.
  • **send_udp** – Prevents the ability to send UDP packets to a specified address by using the sa_send_udp system procedure.
  • **send_email** – Prevents the use of email system procedures, such as xp_sendmail.
  • **web_service_client** – Prevents the use of web service client stored procedure calls (stored procedures that issue HTTP requests).

• **local** – Prevents users from accessing all local-related features. This feature controls access to the server computing environment. This set consists of the **local_call**, **local_db**, **local_io**, and **local_log** feature subsets.

  • **local_call** – Prevents users from accessing all features that provide the ability to execute code that is not directly part of the database server and is not controlled by the database server. This set consists of the following features:

    • **cmdshell** – Prevents the use of the xp_cmdshell procedure.
    • **external_procedure** – Prevents the use of external stored procedures. This setting does not disable the use of the xp_* system procedures (such as xp_cmdshell, xp_readfile, and so on) that are built into the database server. Separate feature control options are provided for these system procedures.
    • **external_procedure_v3** – See the User-Defined Functions guide.
    • **java** – Prevents the use of Java-related features, such as Java procedures.

• **local_db** – Prevents users from accessing all features related to database files. This set consists of the following features:

  • **backup** – Prevents the use of the BACKUP statement, and with it, the ability to run server-side backups. You can still perform client-side backups by using the dbbackup utility.
  • **restore** – Prevents the use of the RESTORE DATABASE statement.
  • **database** – Prevents the use of the CREATE DATABASE, ALTER DATABASE, DROP DATABASE, CREATE ENCRYPTED FILE, CREATE DECRYPTED FILE, CREATE ENCRYPTED DATABASE, and CREATE DECRYPTED DATABASE statements.
• **dbspace** – Prevents the use of the CREATE DBSPACE, ALTER DBSPACE, and DROP DBSPACE statements.

• **local_env** – Prevents users from accessing all features related to environment variables. This set consists of the following features:
  
  • **getenv** – Prevents users from reading the value of any environment variable.

• **local_io** – Prevents users from accessing all features that allow direct access to files and their contents. This set consists of the following features:

  • **create_trace_file** – Prevents the use of statements that create an event tracing target.

  • **read_file** – Prevents the use of statements that can cause a local file to be read. For example, the xp_read_file system procedure, the LOAD TABLE statement, and the use of OPENSTRING( FILE... ). The alternate names load_table and xp_read_file are deprecated.

  • **write_file** – Prevents the use of all statements that can cause a local file to be written to. For example, the UNLOAD statement and the xp_write_file system procedure. The alternate names unload_table and xp_write_file are deprecated.

  • **delete_file** – Prevents the use of all statements that can cause a local file to be deleted. For example, securing this feature causes the dbbackup utility to fail if the -x or -xo options are specified.

  • **directory** – Prevents the use of directory class proxy tables. This feature is disabled when remote_data_access is disabled.

  • **sp_list_directory** – Prevents the use of the sp_list_directory system procedure.

  • **sp_create_directory** – Prevents the use of the sp_create_directory system procedure.

  • **sp_copy_directory** – Prevents the use of the sp_copy_directory system procedure.

  • **sp_move_directory** – Prevents the use of the sp_move_directory system procedure.

  • **sp_delete_directory** – Prevents the use of the sp_delete_directory system procedure.

  • **sp_copy_file** – Prevents the use of the sp_copy_file system procedure.

  • **sp_move_file** – Prevents the use of the sp_move_file system procedure.

  • **sp_delete_file** – Prevents the use of the sp_delete_file system procedure.

• **local_log** – Prevents users from accessing all logging features that result in creating or writing data directly to a file on disk. This set consists of the following features:

  • **request_log** – Prevents the ability to change the request log file name and also prevents the ability to increase the limits of the request log file size or number of files. You can specify the request log file and limits on this file in the command to start the database server; however, they cannot be changed once the database server is started. When request log features are disabled, you can still turn
request logging on and off and reduce the maximum file size and number of request logging files.

- **console_log** – Prevents the ability to change the database server message log file name using the ConsoleLogFile option of the sa_server_option system procedure. Securing this feature also prevents the ability to increase the maximum size of the log file using the ConsoleLogMaxSize option of the sa_server_option system procedure. You can specify a server log file and its size when starting the database server.

- **webclient_log** – Prevents the ability to change the web service client log file name using the WebClientLogFile option of the sa_server_option system procedure. You can specify a web service client log file when starting the database server.

**Applies to**

All operating systems and database servers.

**Remarks**

This option allows the owner of the database server to control whether users have access to features for databases running on the database server. The -sk option allows the owner of the database server to create a system secure feature key that prevents users from accessing features specified by the -sf option.

If you start a database without specifying a system secure feature key, the default secure features are secured, and you cannot change the secure feature settings for the database server or any databases running on it. You cannot create the system secure feature key later—you must shut down the database server and specify a system secure feature key when you restart it.

The *feature-list* is a comma-separated list of feature names or feature sets to secure for the database server. Securing a feature makes it inaccessible to all database users other than administrators. Specifying a feature set secures all the features included in the set. To secure one or more, but not all, of the features in the feature set, specify the individual feature name.

**Note:** Sub-features of feature sets that are secured by default, cannot be unsecured from the command line. In other words the following command will not work:

```
-sf manage_security, -manage_keys
```

Use *feature-name* to indicate that the feature should be secured (made inaccessible), and *-feature-name or feature-name* to indicate that the feature should be unsecured (accessible to all database users). For example, the following command indicates that only dbspace features are accessible to all users:

```
iqsrv16 -n secure_server -sf all,-dbspace
```
Example

The following command starts a database server named secure_server with access to the request log and with all remote data access features secured. The key specified by the -sk option can be used later with the sp_use_secure_feature_key system procedure to make these features accessible to all users on the current connection.

```sql
iqsrv16 -n secure_server -sf remote,-request_log -sk j978kls12
```

If a user connected to a database running on the secure_server database server uses the sp_use_secure_feature_key system procedure with the authorization_key parameter set to the same value as that specified by -sk, that connection has access to the remote data access features:

```sql
CALL sp_use_secure_feature_key ( 'MyKey' , 'j978kls12' );
```

The following command secures all features, with the exception of local database features:

```sql
iqsrv16 -n secure_server -sf all,-local_db
```

Procedures and Functions

Use the system-supplied stored functions and procedures in SAP Sybase IQ databases to retrieve system information.

**sa_get_ldapserver_status System Procedure**

Determines the current status of the LDAP server configuration object.

**Syntax**

```sql
sa_get_ldapserver_status()
```

**Arguments**

None

**Result Set**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldsrv_id</td>
<td>UNSIGNED BIGINT</td>
<td>A unique identifier for the LDAP server configuration object that is the primary key and is used by the login policy to refer to the LDAP server.</td>
</tr>
<tr>
<td>ldsrv_name</td>
<td>CHAR(128)</td>
<td>The name assigned to the LDAP server configuration object.</td>
</tr>
</tbody>
</table>
### Column Name | Data Type | Description
--- | --- | ---
ldsrv_state | CHAR(9) | Read-only state of the LDAP server: 1 – RESET 2 – READY 3 – ACTIVE 4 – FAILED 5 – SUSPENDED A numeric value is stored in system table; a corresponding text value appears in the system view.

### ldsrv_last_state_change | TIMESTAMP | Indicates the time the last state change occurred. The value is stored in Coordinated Universal Time (UTC), regardless of the local time zone of the LDAP server.

**Remarks**
To see SYSLDAPSERVER column values before a checkpoint occurs and the contents of memory are written to the catalog on disk. The updates to the catalog columns ldsrv_state and ldsrv_last_state_change occur asynchronously during checkpoint to the LDAP server object as the result of an event that changes the LDAP server object state, such as a failed connection due to a failed LDAP directory server. The LDAP server object state reflects the state of the LDAP directory server.

**Privileges**
None.

**sa_get_user_status system procedure**
Allows you to determine the current status of users.

**Syntax**

```sql
sa_get_user_status()
```

**Result set**

<table>
<thead>
<tr>
<th>Column name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
<td>UNSIGNED INTEGER</td>
<td>A unique number identifying the user.</td>
</tr>
<tr>
<td>Column name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>user_name</td>
<td>CHAR(128)</td>
<td>The name of the user.</td>
</tr>
<tr>
<td>connections</td>
<td>INTEGER</td>
<td>The current number of connections by this user.</td>
</tr>
<tr>
<td>failed_logins</td>
<td>UNSIGNED INTEGER</td>
<td>The number of failed login attempts made by the user.</td>
</tr>
<tr>
<td>last_login_time</td>
<td>TIMESTAMP</td>
<td>The local time that the user last logged in.</td>
</tr>
<tr>
<td>locked</td>
<td>TINYINT</td>
<td>Indicates if the user account is locked.</td>
</tr>
<tr>
<td>reason_locked</td>
<td>LONG VARCHAR</td>
<td>The reason the account is locked.</td>
</tr>
<tr>
<td>user_dn</td>
<td>CHAR(1024)</td>
<td>The Distinguished Name (DN) for a user ID connecting to an LDAP server.</td>
</tr>
<tr>
<td>user_dn_cached_at</td>
<td>TIMESTAMP</td>
<td>The local time that the DN was stored.</td>
</tr>
<tr>
<td>password_change_state</td>
<td>BIT</td>
<td>A value that indicates whether a dual password change is in progress (0=No, 1=Yes). The default is 0.</td>
</tr>
<tr>
<td>password_change_first_user</td>
<td>UNSIGNED INTEGER</td>
<td>The user_id of the user who set the first part of a dual password; otherwise NULL.</td>
</tr>
<tr>
<td>password_change_second_user</td>
<td>UNSIGNED INTEGER</td>
<td>The user_id of the user who set the second part of a dual password; otherwise NULL.</td>
</tr>
<tr>
<td>user_dn</td>
<td>CHAR(1024)</td>
<td>The distinguished name (DN) of the user.</td>
</tr>
<tr>
<td>user_dn_cached_at</td>
<td>TIMESTAMP</td>
<td>The date and time the distinguished name was found.</td>
</tr>
</tbody>
</table>

**Remarks**

This procedure returns a result set that shows the current status of users. In addition to basic user information, the procedure includes a column indicating if the user has been locked out.
and a column with a reason for the lockout. Users can be locked out for the following reasons: locked due to policy, password expiry, or too many failed attempts.

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.

Privileges
You can view information about yourself; no privilege is required. You must have the MANAGE ANY USER system privilege to view information about other users.

Side effects
None

Example
The following example uses the sa_get_user_status system procedure to return the status of database users.

CALL sa_get_user_status;

sp_create_secure_feature_key System Procedure
Creates a new secure feature key.

Syntax

```
sp_create_secure_feature_key (  
    name,  
    auth_key,  
    features )
```

Arguments

- **name** – the VARCHAR (128) name for the new secure feature key. This argument cannot be NULL or an empty string.
- **auth_key** – the CHAR (128) authorization key for the secure feature key. The authorization key must be a non-empty string of at least six characters.
- **features** – the LONG VARCHAR comma-separated list of secure features that the new key can enable. Specifying ",-" before a feature means that the feature is not re-enabled when the secure feature key is set.

Remarks
This procedure creates a new secure feature key that can be given to any user. The system secure feature key is created using the -sk database server option.

Privileges
To use this procedure, you must be the database server owner and have the manage_keys feature enabled on the connection.
**sp_displayroles System Procedure**

Displays all roles granted to a user-defined role or a user, or displays the entire hierarchical tree of roles.

**Syntax**

dbo.sp_displayroles(
   [user_role_name],
   [display_mode],
   [grant_type] )

**Arguments**

- **user_role_name** – valid values are:
  - A valid system privilege name or system privilege role name
  - A valid user-defined role name
  - A valid user name

  By default, if no argument is specified, the current login user is used.

- **display_mode** – valid values are:
  - **EXPAND_UP** – shows all roles granted the input role or system privilege; that is the role hierarchy tree for the parent levels.
  - **EXPAND_DOWN** – shows all roles or system privileges granted to the input role or user; that is, the role hierarchy tree for the child levels.

  If no argument is specified (default), only the directly granted roles or system privileges appear.

- **grant_type** – valid values are:
  - **ALL** – shows all roles or system privileges granted.
  - **NO_ADMIN** – shows all roles or system privileges granted with the WITH NO ADMIN OPTION or WITH ADMIN OPTION clause.
  - **ADMIN** – shows all roles or system privileges granted with the WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clause.

  If no argument is specified, **ALL** is used.

**Result Set**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>role_name</td>
<td>char(128)</td>
<td>Lists role/system privilege name.</td>
</tr>
<tr>
<td>parent_role_name</td>
<td>char(128)</td>
<td>Lists role name of the parent.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Data Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>grant_type</td>
<td>char(10)</td>
<td>Lists grant type.</td>
</tr>
<tr>
<td>role_level</td>
<td>smallint</td>
<td>For \texttt{Expand_down} mode, 1 indicates directly granted roles; 2 indicates the next hierarchy below, and so on. For \texttt{Expand_up} mode, 0 indicates the roles to which the specified role is granted; -1 indicates the next hierarchy above, and so on.</td>
</tr>
</tbody>
</table>

\textit{Remarks}

For Name = System privilege name, the results show the system privilege name instead of the system privilege role name.

For Mode = Expand\_down, parent\_role\_name is NULL for level 1 (directly granted roles). If no mode is specified (default), role\_level is 1 and parent\_role\_name is NULL, since only directly granted roles appear.

For Name = User name, with Mode = expand\_up, no results are returned since a user resides at the top level in any role hierarchy. Similarly, if Name = an immutable system privilege name, with Mode = Expand\_down, no results are returned because an immutable system privilege resides at the bottom level in any role hierarchy.

For default Mode, parent\_role\_name column is NULL and role\_level is 1.

\textit{Privileges}

For users and extended users:

- No system privilege is required to execute this procedures against themselves.
- MANAGE ROLES system privilege is required to execute this procedure for other users.

For roles and system privileges:

- Administrative privilege over the role or system privilege is required to execute this procedure.
- Role administrators can execute this procedure for roles they administer.

\textit{Example}

This example assumes these GRANT statements have been executed:

```
GRANT SERVER OPERATOR TO r4;
GRANT BACKUP DATABASE TO r3 WITH ADMIN OPTION;
GRANT DROP CONNECTION TO r3 WITH ADMIN ONLY OPTION;
GRANT MONITOR TO r2; GRANT CHECKPOINT TO r1;
GRANT ROLE r2 TO r1 WITH ADMIN OPTION;
GRANT ROLE r3 TO r2 WITH NO ADMIN OPTION;
```
GRANT ROLE r4 TO r3 WITH ADMIN ONLY OPTION;
GRANT ROLE r1 TO user1;
GRANT ROLE r1 TO r7;
GRANT ROLE r7 TO user2 WITH ADMIN OPTION;
GRANT BACKUP DATABASE TO user2 WITH ADMIN ONLY OPTION;

sp_displayroles( 'user2', 'expand_down', 'ALL' ) produces output similar to:

<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>r7</td>
<td>NULL</td>
<td>ADMIN</td>
<td>1</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>NULL</td>
<td>NO ADMIN</td>
<td>1</td>
</tr>
<tr>
<td>BACKUP DATABASE</td>
<td>NULL</td>
<td>ADMIN ONLY</td>
<td>1</td>
</tr>
<tr>
<td>dbo</td>
<td>PUBLIC</td>
<td>NO ADMIN</td>
<td>2</td>
</tr>
<tr>
<td>r1</td>
<td>r7</td>
<td>NO ADMIN</td>
<td>2</td>
</tr>
<tr>
<td>r2</td>
<td>r1</td>
<td>ADMIN</td>
<td>3</td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td>r1</td>
<td>NO ADMIN</td>
<td>3</td>
</tr>
<tr>
<td>r3</td>
<td>r2</td>
<td>NO ADMIN</td>
<td>4</td>
</tr>
<tr>
<td>MONITOR</td>
<td>r2</td>
<td>NO ADMIN</td>
<td>4</td>
</tr>
<tr>
<td>r4</td>
<td>r3</td>
<td>ADMIN ONLY</td>
<td>5</td>
</tr>
<tr>
<td>BACKUP DATABASE</td>
<td>r3</td>
<td>ADMIN</td>
<td>5</td>
</tr>
<tr>
<td>DROP CONNCTION</td>
<td>r3</td>
<td>ADMIN ONLY</td>
<td>5</td>
</tr>
</tbody>
</table>

sp_displayroles( 'user2', 'expand_down', 'NO_ADMIN' ) produces output similar to:

<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>r7</td>
<td>NULL</td>
<td>ADMIN</td>
<td>1</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>NULL</td>
<td>NO ADMIN</td>
<td>1</td>
</tr>
<tr>
<td>dbo</td>
<td>PUBLIC</td>
<td>NO ADMIN</td>
<td>2</td>
</tr>
<tr>
<td>r1</td>
<td>r7</td>
<td>NO ADMIN</td>
<td>2</td>
</tr>
<tr>
<td>r2</td>
<td>r1</td>
<td>ADMIN</td>
<td>3</td>
</tr>
</tbody>
</table>
### sp_displayroles( 'r3', 'expand_up', 'NO_ADMIN' ) produces output similar to:

<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHECKPOINT</td>
<td>r1</td>
<td>NO ADMIN</td>
<td>3</td>
</tr>
<tr>
<td>r3</td>
<td>r2</td>
<td>NO ADMIN</td>
<td>4</td>
</tr>
<tr>
<td>MONITOR</td>
<td>r2</td>
<td>NO ADMIN</td>
<td>4</td>
</tr>
<tr>
<td>BACKUP DATABASE</td>
<td>r3</td>
<td>ADMIN</td>
<td>5</td>
</tr>
</tbody>
</table>

### sp_displayroles( 'r1', 'NO_ADMIN', 'expand_up') produces output similar to:

<table>
<thead>
<tr>
<th>role_name</th>
<th>parent_role_name</th>
<th>grant_type</th>
<th>role_level</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>r7</td>
<td>NO ADMIN</td>
<td>-2</td>
</tr>
<tr>
<td>r2</td>
<td>r1</td>
<td>ADMIN</td>
<td>-1</td>
</tr>
<tr>
<td>r3</td>
<td>r2</td>
<td>NO ADMIN</td>
<td>0</td>
</tr>
</tbody>
</table>

### sp_expireallpasswords system procedure
Immediately expires all user passwords.

**Syntax 1**
call sp_expireallpasswords

**Syntax 2**
sp_expireallpasswords

**Privileges**
Requires the MANAGE ANY USER system privilege.

### SP_HAS_ROLE Function [System]
Returns an integer value indicating whether the invoking user has been granted a specified system privilege or user-defined role. When used for permission checking within user-defined
stored procedures. **SP_HAS_ROLE** returns an error message when a user fails a permission check.

**Syntax**

dbo.sp_has_role( [rolename], [grant_type], [throw_error] )

**Arguments**

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rolename</td>
<td>The name of a system privilege or user-defined role.</td>
</tr>
<tr>
<td>grant_type</td>
<td>Valid values are: ADMIN and NO ADMIN. If NULL or not specified, NO ADMIN is used by default.</td>
</tr>
<tr>
<td>throw_error</td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td>• 1 – display error message specified if system privilege or user-defined role is not granted to invoking user.</td>
</tr>
<tr>
<td></td>
<td>• 0 – (default) do not display error message if specified system privilege or user-defined role is not granted to invoking user.</td>
</tr>
</tbody>
</table>

**Result Set**

<table>
<thead>
<tr>
<th>Value Returned</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System privilege or user-defined role is granted to invoking user.</td>
</tr>
<tr>
<td>0 or Permission denied: you do not have permission to execute this command/procedure.</td>
<td>System privilege or user-defined role is not granted to invoking user. The error message replaces the value 0 when the throw_error argument is set to 1.</td>
</tr>
<tr>
<td>-1</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Remarks**

If the value of the **grant_type** argument is ADMIN, the function checks whether the invoking user has administrative privileges for the system privilege. If the value of the **grant_type** argument is NO ADMIN, the function checks whether the invoking user has privileged use of the system privilege or role.

If the **grant_type** argument is not specified, NO ADMIN is used by default and output indicates only whether the invoking user has been granted, either directly or indirectly, the specified system privilege or user-defined role.
If the rolename and grant_type arguments are both NULL and the throw_error argument is 1, you see an error message. You may find this useful for those stored procedures where an error message appears after certain values are read from the catalog tables rather than after the checking the presence of certain system privileges for the invoking user.

**Note:** A permission denied error message is returned if the arguments rolename and grant_type are set to NULL and throw_error is set to 1, or if all three arguments are set to NULL.

**Permissions**
None

**Example 1**
Consider the following scenario:

- u1 has been granted the CREATE ANY PROCEDURE system privilege with the WITH NO ADMIN OPTION clause.
- u1 has not been granted the CREATE ANY TABLE system privilege.
- u1 has been granted the user-defined role Role_A with the WITH ADMIN ONLY OPTION clause.
- Role_B exists, but has not been granted to u1
- The role Role_C does not exist.

Based on the above scenario, this command

- `sp_has_role 'create any procedure'`
  returns the value 1, which indicates u1 has been granted the CREATE ANY PROCEDURE system privilege.
- `sp_has_role 'create any table'`
  returns the value 0, which indicates u1 has not been granted the CREATE ANY TABLE system privilege. No error message is returned because the throw_error argument is not specified.
- `sp_has_role 'create any procedure','admin',1`
  returns the Permission denied error message (throw_error=1). Even though u1 has been granted the CREATE ANY PROCEDURE system privilege, u1 has not been granted administrative rights to the system privilege.
- `sp_has_role 'Role_A'`
  returns the value 1, which indicates u1 has been granted role Role_A.
- `sp_has_role 'Role_A','admin',1`
  returns the value 1, which indicates u1 has been granted role Role_A with administrative rights.
- `sp_has_role 'Role_B'`
returns the value 0, which indicates u1 has not been granted the role ROLE_B. No error message is returned because the throw_error argument is not specified.

- **sp_has_role 'Role_C'**

returns the value -1, which indicates the role ROLE_C does not exist.

- **sp_has_role 'Role_C',NULL,1**

returns the value -1, which indicates the role ROLE_C does not exist.

**sp_iqaddlogin Procedure**

Adds a new SAP Sybase IQ user account to the specified login policy.

**Syntax1**

```sql
call sp_iqaddlogin ('username_in', 'pwd', ['password_expiry_on_next_login'] [ , 'policy_name' ] )
```

**Syntax2**

```sql
sp_iqaddlogin 'username_in', 'pwd', ['password_expiry_on_next_login'] [ , 'policy_name' ]
```

**Syntax3**

```sql
sp_iqaddlogin username_in, pwd, [ password_expiry_on_next_login ] [ , policy_name ]
```

**Usage**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username_in</td>
<td>The user’s login name. Login names must conform to the rules for identifiers</td>
</tr>
<tr>
<td>pwd</td>
<td>The user’s password. Passwords must conform to rules for passwords, that is, they must be valid identifiers.</td>
</tr>
<tr>
<td>password_expiry_on_next_login</td>
<td>(Optional) Specifies whether user’s password expires as soon as this user’s login is created. Default setting is OFF (password does not expire).</td>
</tr>
<tr>
<td>policy_name</td>
<td>(Optional) Creates the user under the named login policy. If unspecified, user is created under the root login policy.</td>
</tr>
</tbody>
</table>

A username_in/pwd created using `sp_iqaddlogin` and set to expire in one day is valid all day tomorrow and invalid on the following day. In other words, a login created today and set to expire in \( n \) days are not usable once the date changes to the \( (n+1) \)th day.
Privilages
Requires the MANAGE ANY USER system privilege.

Description
Adds a new SAP Sybase IQ user account, assigns a login policy to the user and adds the user to the ISYSUSER system table. If the user already has a user ID for the database but is not in ISYSUSER, (for example, if the user ID was added using the GRANT CONNECT statement or Sybase Control Center), \texttt{sp_iqaddlogin} adds the user to the table.

If you do not specify a login policy name when calling the procedure, SAP Sybase IQ assigns the user to the root login policy.

\textbf{Note:} If the maximum number of logins for a login policy is unlimited, then a user belonging to that login policy can have an unlimited number of connections.

The first user login forces a password change and assigns a login policy to the newly created user. Use \texttt{CREATE USER} to create new users, although, for backward compatibility, \texttt{sp_iqaddlogin} is still supported.

Examples
These calls add the user \texttt{rose} with a password \texttt{irk324} under the login policy named \texttt{expired_password}. This example assumes the \texttt{expired_password} login policy already exists.

\begin{verbatim}
call sp_iqaddlogin('rose', 'irk324', 'ON', 'expired_password')
sp_iqaddlogin 'rose','irk324', 'ON', 'expired_password'
\end{verbatim}

\textbf{sp_iqbackupdetails Procedure}
Shows all the dbfiles included in a particular backup.

\textbf{Syntax}
\begin{verbatim}
sp_iqbackupdetails backup_id
\end{verbatim}

\textbf{Parameters}

\begin{table}[h]
\centering
\begin{tabular}{|l|p{10cm}|}
\hline
\textbf{Parameter} & \textbf{Description} \\
\hline
backup_id & Specifies the backup operation transaction identifier. \\
\hline
\end{tabular}
\end{table}

\textbf{Note:} You can obtain the backup_id value from the SYSIQBACKUPHISTORY table by executing the query:
\begin{verbatim}
select * from sysiqbackuphistory
\end{verbatim}
Privileges
No specific system privileges are required to run this procedure.

Description
`sp_iqbackupdetails` returns:

Table 12. sp_iqbackupdetails Columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup_id</td>
<td>Identifier for the backup transaction.</td>
</tr>
<tr>
<td>backup_time</td>
<td>Time of the backup.</td>
</tr>
<tr>
<td>backup_type</td>
<td>Type of backup: “Full,” “Incremental since incremental,” or “Incremental since full.”</td>
</tr>
<tr>
<td>depends_on_id</td>
<td>Identifier for previous backup that the backup depends on.</td>
</tr>
<tr>
<td>dbspace_id</td>
<td>Identifier for the dbspace being backed up.</td>
</tr>
<tr>
<td>dbspace_name</td>
<td>Name of the dbspace from SYSIQBACKUPHISTORYDETAIL. If dbspace name matches the dbspace name in SYSDBSPACE for a given dbspace_id. Otherwise “null.”</td>
</tr>
<tr>
<td>dbspace_rwstatus</td>
<td>“ReadWrite” or “Read Only.”</td>
</tr>
<tr>
<td>dbspace_createid</td>
<td>Dbspace creation transaction identifier.</td>
</tr>
<tr>
<td>dbspace_alterid</td>
<td>Alter DBSPACE read-write mode transaction identifier.</td>
</tr>
<tr>
<td>dbspace_online</td>
<td>Status “Online” or “Offline.”</td>
</tr>
<tr>
<td>dbspace_size</td>
<td>Size of dbspace, in KB, at time of backup.</td>
</tr>
<tr>
<td>dbspace_backup_size</td>
<td>Size of data, in KB, backed up in the dbspace.</td>
</tr>
<tr>
<td>dbfile_id</td>
<td>Identifier for the dbfile being backed up.</td>
</tr>
<tr>
<td>dbfile_name</td>
<td>The logical file name, if it was not renamed after the backup operation. If renamed, “null.”</td>
</tr>
<tr>
<td>dbfile_rwstatus</td>
<td>“ReadWrite” or “Read Only.”</td>
</tr>
<tr>
<td>dbfile_createid</td>
<td>Dbfile creation transaction identifier.</td>
</tr>
<tr>
<td>dbfile_alterid</td>
<td>Alter DBSPACE alter FILE read-write mode transaction identifier.</td>
</tr>
</tbody>
</table>
### Column Name

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbfile_size in MB</td>
<td>Size of the dbfile, in KB.</td>
</tr>
<tr>
<td>dbfile_backup_size</td>
<td>Size of the dbfile backup, in KB.</td>
</tr>
<tr>
<td>dbfile_path</td>
<td>The dbfile path from SYSBACKUPDETAIL, if it matches the physical file path</td>
</tr>
<tr>
<td></td>
<td>(“file_name”) in SYSDBFILE for a given dbspace_id and the dbfile_id. Other-</td>
</tr>
<tr>
<td></td>
<td>wise “null.”</td>
</tr>
</tbody>
</table>

### Example

Sample output from **sp_iqbackupdetails**:

```
backup_id   backup_time             backup_type   selective_type   depends_on_id
depends_on_id
883   2008-09-23 13:58:49.0   Full          All                0

dbspace_id   dbspace_name   dbspace_rwstatus   dbspace_createid
dbspace_alterid   dbspace_online   dbspace_size   dbspace_backup_size
dbfile_id
0                0         2884                2884         0

dbfile_name dbfile_rwstatus dbfile_createid dbfile_alterid
system        ReadWrite                 0               0       2884

dbfile_size
system        ReadWrite                 0               0       2884

dbfile_backup_size dbfile_path
2884  C:\Documents and Settings\All Users\SybaseIQ\demo\iqdemo.db
```

### sp_iqbackupsummary Procedure

Summarizes backup operations performed.

### Syntax

```
sp_iqbackupsummary [ timestamp or backup_id ]
```

### Parameters

- **timestamp or backup_id** – specifies the interval for which to report backup operations. If you specify a timestamp or a backup ID, only those records with backup_time greater than or equal to the time you enter are returned. If you specify no timestamp, the procedure returns all the backup records in `ISYSIQBACKUPHISTORY`.

### Privileges

No specific system privileges are required to run this procedure.
Description

Table 13. sp_iqbackupsummary Columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backup_id</td>
<td>Identifier for the backup transaction</td>
</tr>
<tr>
<td>backup_time</td>
<td>Time of the backup</td>
</tr>
<tr>
<td>backup_type</td>
<td>Type of backup: “Full,” “Incremental since incremental,” or “Incremental since full”</td>
</tr>
<tr>
<td>selective_type</td>
<td>Subtype of backup: &quot;All Inclusive,&quot; &quot;All RW files in RW dbspaces,&quot; &quot;Set of RO dbspace/file&quot;</td>
</tr>
<tr>
<td>virtual_type</td>
<td>Type of virtual backup: “Non-virtual,” “Decoupled,” or “Encapsulated”</td>
</tr>
<tr>
<td>depends_on_id</td>
<td>Identifier for backup that the backup depends on</td>
</tr>
<tr>
<td>creator</td>
<td>Creator of the backup</td>
</tr>
<tr>
<td>backup_size</td>
<td>Size, in KB, of the backup</td>
</tr>
<tr>
<td>user_comment</td>
<td>User comment</td>
</tr>
<tr>
<td>backup_command</td>
<td>The backup statement issued (minus the comment)</td>
</tr>
</tbody>
</table>

Example
Sample output of sp_iqbackupsummary:

```
backup_id   backup_time             backup_type   selective_type   virtual_type
883   2008-09-23 13:58:49.0   Full          All inclusive    Non virtual
depends_on_id   creator   backup_size   user_comment   backup_command
0   DBA                10864               backup database to 'c:\\temp\\b1'
```

sp_iqconnection Procedure
Shows information about connections and versions, including which users are using temporary dbspace, which users are keeping versions alive, what the connections are doing inside SAP Sybase IQ, connection status, database version status, and so on.

Syntax

```sql
sp_iqconnection [ connhandle ]
```
Applies to
Simplex and multiplex.

Usage
connhandle is equal to the Number connection property and is the ID number of the connection. The connection_property system function returns the connection ID:

```
SELECT connection_property ( 'Number' )
```

When called with an input parameter of a valid connhandle, sp_iqconnection returns the one row for that connection only.

Privileges
Requires the DROP CONNECTION, MONITOR or SERVER OPERATOR system privilege. Users without one of these system privileges must be granted EXECUTE permission to run the stored procedure.

Description
sp_iqconnection returns a row for each active connection. The columns ConnHandle, Name, Userid, LastReqTime, ReqType, CommLink, NodeAddr, and LastIdle are the connection properties Number, Name, Userid, LastReqTime, ReqType, CommLink, NodeAddr, and LastIdle respectively, and return the same values as the system function sa_conn_info. The additional columns return connection data from the SAP Sybase IQ side of the SAP Sybase IQ engine. Rows are ordered by ConnCreateTime.

The column MPXServerName stores information related to internode communication (INC), as shown:

<table>
<thead>
<tr>
<th>Server Where Run</th>
<th>MPXServerName Column Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplex server</td>
<td>NULL (All connections are local/user connections)</td>
</tr>
<tr>
<td>Multiplex coordinator</td>
<td>• NULL for local/user connections</td>
</tr>
<tr>
<td></td>
<td>• Contains value of secondary node’s server name (source of connection) for every INC</td>
</tr>
<tr>
<td></td>
<td>connection (either on-demand or dedicated heartbeat connection)</td>
</tr>
<tr>
<td>Multiplex secondary</td>
<td>• NULL for local/user connections</td>
</tr>
<tr>
<td></td>
<td>• Contains value of coordinator’s server name (source of connection).</td>
</tr>
</tbody>
</table>
In Java applications, specify SAP Sybase IQ-specific connection properties from TDS clients in the RemotePWD field. This example, where `myconnection` becomes the IQ connection name, shows how to specify IQ specific connection parameters:

```java
p.put("RemotePWD","",CON=myconnection");
```

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnHandle</td>
<td>The ID number of the connection.</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the server.</td>
</tr>
<tr>
<td>Userid</td>
<td>The user ID for the connection.</td>
</tr>
<tr>
<td>LastReqTime</td>
<td>The time at which the last request for the specified connection started.</td>
</tr>
<tr>
<td>ReqType</td>
<td>A string for the type of the last request.</td>
</tr>
<tr>
<td>IQCmdType</td>
<td>The current command executing on the SAP Sybase IQ side, if any. The command type reflects commands defined at the implementation level of the engine. These commands consist of transaction commands, DDL and DML commands for data in the IQ store, internal IQ cursor commands, and special control commands such as OPEN and CLOSE DB, BACKUP, RESTORE, and others.</td>
</tr>
<tr>
<td>LastIQCmdTime</td>
<td>The time the last IQ command started or completed on the IQ side of the SAP Sybase IQ engine on this connection.</td>
</tr>
<tr>
<td>IQCursors</td>
<td>The number of cursors open in the IQ store on this connection.</td>
</tr>
<tr>
<td>LowestIQCursorState</td>
<td>The IQ cursor state, if any. If multiple cursors exist on the connection, the state that appears is the lowest cursor state of all the cursors; that is, the furthest from completion. Cursor state reflects internal SAP Sybase IQ implementation detail and is subject to change in the future. For this version, cursor states are: NONE, INITIALIZED, PARSED, DESCRIBED, COSTED, PREPARED, EXECUTED, FETCHING, END_OF_DATA, CLOSED and COMPLETED. As suggested by the names, cursor state changes at the end of the operation. A state of PREPARED, for example, indicates that the cursor is executing.</td>
</tr>
<tr>
<td>IQthreads</td>
<td>The number of SAP Sybase IQ threads currently assigned to the connection. Some threads may be assigned but idle. This column can help you determine which connections are using the most resources.</td>
</tr>
<tr>
<td>TxnID</td>
<td>The transaction ID of the current transaction on the connection. This is the same as the transaction ID in the .iqmsg file by the BeginTxn, CmtTxn, and PostCmtTxn messages, as well as the Txn ID Seq logged when the database is opened.</td>
</tr>
<tr>
<td>ConnCreateTime</td>
<td>The time the connection was created.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TempTableSpaceKB</td>
<td>The number of kilobytes of IQ temporary store space in use by this connection for data stored in IQ temp tables.</td>
</tr>
<tr>
<td>TempWorkSpaceKB</td>
<td>The number of kilobytes of IQ temporary store space in use by this connection for working space such as sorts, hashes, and temporary bitmaps. Space used by bitmaps or other objects that are part of indexes on SAP Sybase IQ temporary tables are reflected in TempTableSpaceKB.</td>
</tr>
<tr>
<td>IQConnID</td>
<td>The ten-digit connection ID included as part of all messages in the .iqmsg file. This is a monotonically increasing integer unique within a server session.</td>
</tr>
<tr>
<td>satoiq_count</td>
<td>An internal counter used to display the number of crossings from the SQL Anywhere side to the IQ side of the SAP Sybase IQ engine. This might be occasionally useful in determining connection activity. Result sets are returned in buffers of rows and do not increment satoiq_count or iqtosa_count once per row.</td>
</tr>
<tr>
<td>iqtosa_count</td>
<td>An internal counter used to display the number of crossings from the IQ side to the SQL Anywhere side of the SAP Sybase IQ engine. This might be occasionally useful in determining connection activity.</td>
</tr>
<tr>
<td>CommLink</td>
<td>The communication link for the connection. This is one of the network protocols supported by SAP Sybase IQ, or is local for a same-machine connection.</td>
</tr>
<tr>
<td>NodeAddr</td>
<td>The node for the client in a client/server connection.</td>
</tr>
<tr>
<td>LastIdle</td>
<td>The number of ticks between requests.</td>
</tr>
<tr>
<td>MPXServerName</td>
<td>If an INC connection, the varchar(128) value contains the name of the multiplex server where the INC connection originates. NULL if not an INC connection.</td>
</tr>
<tr>
<td>LSName</td>
<td>The logical server name of the connection. NULL if logical server context is unknown or not applicable.</td>
</tr>
<tr>
<td>INCConnName</td>
<td>The name of the underlying INC connection for a user connection. The data type for this column is varchar(255). If <strong>sp_iqconnection</strong> shows an INC connection name for a suspended user connection, that user connection has an associated INC connection that is also suspended.</td>
</tr>
<tr>
<td>INCConnSuspended</td>
<td>The value &quot;Y&quot; in this column indicates that the underlying INC connection for a user connection is in a suspended state. The value &quot;N&quot; indicates that the connection is not suspended.</td>
</tr>
</tbody>
</table>

**Example**

**sp_iqconnection**

<table>
<thead>
<tr>
<th>ConnHandle</th>
<th>Name</th>
<th>Userid</th>
<th>LastReqTime</th>
<th>ReqType</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>'SQL_DBC_100525210'</td>
<td>'DBA'</td>
<td>'2011-03-28 09:29:24.466'</td>
<td>'OPEN'</td>
</tr>
</tbody>
</table>
**sp_iqcopyloginpolicy Procedure**

Creates a new login policy by copying an existing one.

**Syntax1**

```sql
call sp_iqcopyloginpolicy ('existing-policy-name', 'new-policy-name')
```

**Syntax2**

```sql
sp_iqcopyloginpolicy 'existing-policy-name', 'new-policy-name'
```

**Usage**

**Table 14. Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>existing policy name</td>
<td>The login policy to copy.</td>
</tr>
<tr>
<td>new policy name</td>
<td>Name of the new login policy to create (CHAR(128)).</td>
</tr>
</tbody>
</table>

**Privileges**

Requires the MANAGE ANY LOGIN POLICY system privilege.

**Examples**

Creates a new login policy named `lockeduser` by copying the login policy option values from the existing login policy named `root`:

```sql
call sp_iqcopyloginpolicy ('root','lockeduser')
```
**sp_iqdbspace Procedure**
Displays detailed information about each IQ dbspace.

**Syntax**
```
sp_iqdbspace [ dbspace-name ]
```

**Applies to**
Simplex and multiplex.

**Privileges**
Requires MANAGE ANY DBSPACE system privilege. Users without MANAGE ANY DBSPACE system privilege must be granted EXECUTE permission.

**Description**
Use the information from `sp_iqdbspace` to determine whether data must be moved, and for data that has been moved, whether the old versions have been deallocated.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSpaceName</td>
<td>Name of the dbspace as specified in the CREATE DBSPACE statement. Dbspace names are always case-insensitive, regardless of the CREATE DATABASE...CASE IGNORE or CASE RESPECT specification.</td>
</tr>
<tr>
<td>DBSpaceType</td>
<td>Type of the dbspace (MAIN, SHARED_TEMP, TEMPORARY, or RLV).</td>
</tr>
<tr>
<td>Writable</td>
<td>T (writable) or F (not writable).</td>
</tr>
<tr>
<td>Online</td>
<td>T (online) or F (offline).</td>
</tr>
<tr>
<td>Usage</td>
<td>Percent of dbspace currently in use by all files in the dbspace.</td>
</tr>
<tr>
<td>TotalSize</td>
<td>Total size of all files in the dbspace in the units B (bytes), K (kilobytes), M (megabytes), G (gigabytes), T (terabytes), or P (petabytes).</td>
</tr>
<tr>
<td>Reserve</td>
<td>Total reserved space that can be added to all files in the dbspace.</td>
</tr>
<tr>
<td>NumFiles</td>
<td>Number of files in the dbspace.</td>
</tr>
<tr>
<td>NumRWFiles</td>
<td>Number of read/write files in the dbspace.</td>
</tr>
<tr>
<td>Stripingon</td>
<td>F (Off).</td>
</tr>
<tr>
<td>StripeSize</td>
<td>Always 1, if disk striping is on.</td>
</tr>
<tr>
<td>BlkTypes</td>
<td>Space used by both user data and internal system structures.</td>
</tr>
<tr>
<td>OKToDrop</td>
<td>&quot;Y&quot; indicates the dbspace can be dropped; otherwise &quot;N&quot;.</td>
</tr>
</tbody>
</table>

Values of the BlkTypes block type identifiers:
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Block Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Active version</td>
</tr>
<tr>
<td>B</td>
<td>Backup structures</td>
</tr>
<tr>
<td>C</td>
<td>Checkpoint log</td>
</tr>
<tr>
<td>D</td>
<td>Database identity</td>
</tr>
<tr>
<td>F</td>
<td>Free list</td>
</tr>
<tr>
<td>G</td>
<td>Global free list manager</td>
</tr>
<tr>
<td>H</td>
<td>Header blocks of the free list</td>
</tr>
<tr>
<td>I</td>
<td>Index advice storage</td>
</tr>
<tr>
<td>M</td>
<td>Multiplex CM*</td>
</tr>
<tr>
<td>O</td>
<td>Old version</td>
</tr>
<tr>
<td>R</td>
<td>RLV free list manager</td>
</tr>
<tr>
<td>T</td>
<td>Table use</td>
</tr>
<tr>
<td>U</td>
<td>Index use</td>
</tr>
<tr>
<td>N</td>
<td>Column use</td>
</tr>
<tr>
<td>X</td>
<td>Drop at checkpoint</td>
</tr>
</tbody>
</table>

*The multiplex commit identity block (actually 128 blocks) exists in all IQ databases, even though it is not used by simplex databases.

**Example**

Displays information about dbspaces:

```
sp_iqdbspace;
```

**Note:** The following example shows objects in the `iqdemo` database to better illustrate output. `iqdemo` includes a sample user dbspace named `iq_main` that may not be present in your own databases.
## sp_iqdbspaceinfo Procedure

Displays the size of each object and subobject used in the specified table. Not supported for RLV dbspaces.

### Syntax

```sql
sp_iqdbspaceinfo [ dbspace-name ] [, owner_name ] [, object_name ] [, object-type ]
```

### Applies to

Simplex and multiplex.

### Privileges

Requires the BACKUP DATABASE, SERVER OPERATOR, or MANAGE ANY DBSPACE system privileges. Users without one of these system privileges must be granted EXECUTE permission.
Usage

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbspace_name</td>
<td>If specified, sp_iqdbspaceinfo displays one line for each table that has any component in the specified dbspace. Otherwise, the procedure shows information for all dbspaces in the database.</td>
</tr>
<tr>
<td>owner_name</td>
<td>Owner of the object. If specified, sp_iqdbspaceinfo displays output only for tables with the specified owner. If not specified, sp_iqdbspaceinfo displays information on tables for all users in the database.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the table. If not specified, sp_iqdbspaceinfo displays information on all tables in the database.</td>
</tr>
<tr>
<td>object_type</td>
<td>Valid table objects.</td>
</tr>
</tbody>
</table>

All parameters are optional, and any parameter may be supplied independent of another parameter’s value.

The sp_iqdbspaceinfo stored procedure supports wildcard characters for interpreting dbspace_name, object_name, and owner_name. It shows information for all dbspaces that match the given pattern in the same way the LIKE clause matches patterns inside queries.

Description

The procedure returns no results if you specify an RLV dbspace.

sp_iqdbspaceinfo shows the DBA the amount of space used by objects that reside on each dbspace. The DBA can use this information to determine which objects must be relocated before a dbspace can be dropped. The subobject columns display sizes reported in integer quantities followed by the suffix B, K, M, G, T, or P, representing bytes, kilobytes, megabytes, gigabytes, terabytes, and petabytes, respectively.

For tables, sp_iqdbspaceinfo displays subobject sizing information for all subobjects (using integer quantities with the suffix B, K, M, G, T, or P) sorted by dbspace_name, object_name, and owner_name.

Table 15. sp_iqdbspaceinfo Columns

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbspace_name</td>
<td>Name of the dbspace.</td>
</tr>
<tr>
<td>Column Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>object_type</td>
<td>Type of the object (table or joinindex only).</td>
</tr>
<tr>
<td>owner</td>
<td>Name of the owner of the object.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the object on the dbspace.</td>
</tr>
<tr>
<td>object_id</td>
<td>Global object ID of the object.</td>
</tr>
<tr>
<td>id</td>
<td>Table id of the object.</td>
</tr>
<tr>
<td>columns</td>
<td>Size of column storage space on the given dbspace.</td>
</tr>
<tr>
<td>indexes</td>
<td>Size of index storage space on the given dbspace. Does not use system-</td>
</tr>
<tr>
<td></td>
<td>generated indexes (for example, HG indexes in unique constraints or FP</td>
</tr>
<tr>
<td></td>
<td>indexes).</td>
</tr>
<tr>
<td>metadata</td>
<td>Size of storage space for metadata objects on the given dbspace.</td>
</tr>
<tr>
<td>primary_key</td>
<td>Size of storage space for primary key related objects on the given</td>
</tr>
<tr>
<td></td>
<td>dbspace.</td>
</tr>
<tr>
<td>unique_constraint</td>
<td>Size of storage space for unique constraint-related objects on the given</td>
</tr>
<tr>
<td></td>
<td>dbspace.</td>
</tr>
<tr>
<td>foreign_key</td>
<td>Size of storage space for foreign-key-related objects on the given</td>
</tr>
<tr>
<td></td>
<td>dbspace.</td>
</tr>
<tr>
<td>dbspace_online</td>
<td>Indicates if the dbspace is online (Y) or offline (N).</td>
</tr>
</tbody>
</table>

If you run sp_iqdbspaceinfo against a server you have started with the -r switch (read-only), you see the error Msg 13768, Level 14, State 0: SQL Anywhere Error -757: Modifications not permitted for read-only database. This behavior is expected. The error does not occur on other stored procedures such as sp_iqdbspace, sp_iqfile, sp_iqdbspaceobjectinfo or sp_iqobjectinfo.

**Examples**

**Note:** These examples show objects in the iqdemo database to better illustrate output. iqdemo includes a sample user dbspace named iq_main that may not be present in your own databases.

Displays the size of all objects and subobjects in all tables in all dbspaces in the database:

```
sp_iqdbspaceinfo
```

<table>
<thead>
<tr>
<th>dbspace_name</th>
<th>object_type</th>
<th>owner</th>
<th>object_name</th>
<th>object_id</th>
<th>id</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>iq_main</td>
<td>table</td>
<td>DBA</td>
<td>empl</td>
<td>3689</td>
<td>741</td>
<td>96K</td>
</tr>
<tr>
<td>iq_main</td>
<td>table</td>
<td>DBA</td>
<td>iq_dummy</td>
<td>3686</td>
<td>740</td>
<td>24K</td>
</tr>
<tr>
<td>iq_main</td>
<td>table</td>
<td>DBA</td>
<td>sale</td>
<td>3698</td>
<td>742</td>
<td>96K</td>
</tr>
<tr>
<td>iq_main</td>
<td>table</td>
<td>GROUPO</td>
<td>Contacts</td>
<td>3538</td>
<td>732</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix: SQL Reference

Displays the size of all objects and subobjects owned by a specified user in a specified dbspace in the database:

```sql
sp_iqdbspaceinfo iq_main,GROUPO
```

<table>
<thead>
<tr>
<th>object_type</th>
<th>owner</th>
<th>object_name</th>
<th>object_id</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>Contacts</td>
<td>3538</td>
<td>732</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>Customers</td>
<td>3515</td>
<td>731</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>Departments</td>
<td>3632</td>
<td>738</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>Employees</td>
<td>3641</td>
<td>739</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>FinancialCodes</td>
<td>3612</td>
<td>736</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>FinancialData</td>
<td>3621</td>
<td>737</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>Products</td>
<td>3593</td>
<td>735</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>SalesOrderItems</td>
<td>3580</td>
<td>734</td>
</tr>
<tr>
<td>table</td>
<td>GROUPO</td>
<td>SalesOrders</td>
<td>3565</td>
<td>733</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>indexes</th>
<th>metadata</th>
<th>primary_key</th>
<th>unique_constraint</th>
<th>foreign_key</th>
<th>dbspace_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B</td>
<td>1.37M</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>464K</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>1.22M</td>
<td>0B</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>5.45M</td>
<td>24K</td>
<td>0B</td>
<td>48K</td>
<td>iq_main</td>
</tr>
<tr>
<td>48K</td>
<td>4.63M</td>
<td>24K</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>1.78M</td>
<td>24K</td>
<td>0B</td>
<td>48K</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>8.03M</td>
<td>24K</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>1.53M</td>
<td>24K</td>
<td>0B</td>
<td>48K</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>2.19M</td>
<td>24K</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>192K</td>
<td>4.67M</td>
<td>24K</td>
<td>0B</td>
<td>0B</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>2.7M</td>
<td>24K</td>
<td>0B</td>
<td>104K</td>
<td>iq_main</td>
</tr>
<tr>
<td>0B</td>
<td>3.35M</td>
<td>24K</td>
<td>0B</td>
<td>144K</td>
<td>iq_main</td>
</tr>
</tbody>
</table>
Displays the size of a specified object and its subobjects owned by a specified user in a specified dbspace in the database:

```
sp_iqdbspaceinfo iq_main,GROUPO,Departments
```

<table>
<thead>
<tr>
<th>dbspace_name</th>
<th>object_type</th>
<th>owner</th>
<th>object_name</th>
<th>object_id</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>iq_main</td>
<td>table</td>
<td>GROUPO</td>
<td>Departments</td>
<td>3632</td>
<td>738</td>
</tr>
</tbody>
</table>

```
sp_iqdbspaceobjectinfo Procedure
```

Lists objects and subobjects of type table (including columns, indexes, metadata, primary keys, unique constraints, foreign keys, and partitions) for a given dbspace. Not supported for RLV dbspaces.

**Syntax**

```
sp_iqdbspaceobjectinfo [ dbspace-name ] [ , owner_name ] [ ,
object_name ] [ , object-type ]
```

**Privileges**

No specific system privilege required.

**Usage**

All parameters are optional and any parameter may be supplied independent of the value of other parameters.

**Table 16. Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbspace-name</td>
<td>If specified, <code>sp_iqdbspaceobjectinfo</code> displays output only for the specified dbspace. Otherwise, it shows information for all dbspaces in the database.</td>
</tr>
</tbody>
</table>
The `sp_iqdbspaceobjectinfo` stored procedure supports wildcard characters for interpreting `dbspace_name`, `object_name`, and `owner_name`. It displays information for all dbspaces that match the given pattern in the same way as the `LIKE` clause matches patterns inside queries.

Description
The procedure returns no results if you specify an RLV dbspace.

For tables, `sp_iqdbspaceobjectinfo` displays summary information for all associated subobjects sorted by `dbspace_name`, `owner` and `object_name`.

`sp_iqdbspaceobjectinfo` displays the following information, based on the input parameter values:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbspace_name</td>
<td>Name of the dbspace.</td>
</tr>
<tr>
<td>dbspace_id</td>
<td>Identifier of the dbspace.</td>
</tr>
<tr>
<td>object_type</td>
<td>Table.</td>
</tr>
<tr>
<td>owner</td>
<td>Name of the owner of the object.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the table object on the dbspace.</td>
</tr>
<tr>
<td>object_id</td>
<td>Global object ID of the object.</td>
</tr>
<tr>
<td>id</td>
<td>Table ID of the object.</td>
</tr>
<tr>
<td>columns</td>
<td>Number of table columns which are located on the given dbspace. If a column or one of the column-partitions is located on a dbspace, it is counted to be present on that dbspace. The result is shown in the form n/N (n out of total N columns of the table are on the given dbspace).</td>
</tr>
</tbody>
</table>
### Column Name | Description
--- | ---
indexes | Number of user-defined indexes on the table which are located on the given dbspace. Shown in the form n/N (n out of total N indexes on the table are on the given dbspace). This does not contain indexes which are system-generated, such as FP indexes and HG indexes in the case of unique constraints.
metadata | Boolean field (Y/N) that denotes whether the metadata information of the subobject is also located on this dbspace.
primary_key | Boolean field (1/0) that denotes whether the primary key of the table, if any, is located on this dbspace.
unique_constraint | Number of unique constraints on the table that are located on the given dbspace. Appears in the form n/N (n out of total N unique constraints on the table are in the given dbspace).
foreign_key | Number of foreign_keys on the table that are located on the given dbspace. Appears in the form n/N (n out of total N foreign keys on the table are in the given dbspace).
partitions | Number of partitions of the table that are located on the given dbspace. Appears in the form n/N (n out of total N partitions of the table are in the given dbspace).

### Examples
These examples show objects in the iqdemo database to better illustrate output. iqdemo includes a sample user dbspace named iq_main that may not be present in your own databases.

Displays information about a specific dbspace in the database:

```
sp_iqdbspaceobjectinfo iq_main
```

<table>
<thead>
<tr>
<th>dbspace_name</th>
<th>dbspace_id</th>
<th>object_type</th>
<th>owner</th>
<th>object_name</th>
<th>object_iid</th>
</tr>
</thead>
<tbody>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>DBA</td>
<td>emp1</td>
<td>3689</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>DBA</td>
<td>iq_dummy</td>
<td>3686</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>DBA</td>
<td>sale</td>
<td>3698</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>DBA</td>
<td>Contacts</td>
<td>3538</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>GROUPO</td>
<td>Customers</td>
<td>3515</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>GROUPO</td>
<td>Departments</td>
<td>3632</td>
</tr>
<tr>
<td>iq_main</td>
<td>16387</td>
<td>table</td>
<td>GROUPO</td>
<td>Employees</td>
<td>3641</td>
</tr>
</tbody>
</table>
Displays information about the objects owned by a specific user in a specific dbspace in the database:

```
sp_iqdbspaceobjectinfo iq_main,GROUPO
```

dbdbase_name  dbspace_id  object_type  owner  object_name  object_id  columns
iq_main  16387  table  GROUPO  Contacts  3538
iq_main  16387  table  GROUPO  Customers  3515
iq_main  16387  table  GROUPO  Departments  3632
iq_main  16387  table  GROUPO  Employees  3641
iq_main  16387  table  GROUPO  FinancialCodes  3612
iq_main  16387  table  GROUPO  FinancialData  3621
iq_main  16387  table  GROUPO  Products  3593
iq_main  16387  table  GROUPO  SalesOrderItems  3580
iq_main  16387  table  GROUPO  SalesOrders  3565

indexes  metadata  primary_key  unique_constraint  foreign_key  partitions
0/0  Y  0  0/0  0/0  0/0
0/0  Y  0  0/0  0/0  0/0
0/0  Y  0  0/0  0/0  0/0
0/0  Y  1  0/0  1/1  0/0
1/1  Y  1  0/0  0/0  0/0
0/0  Y  1  0/0  1/1  0/0
0/0  Y  1  0/0  0/0  0/0
0/0  Y  1  0/0  1/1  0/0
0/0  Y  1  0/0  0/0  0/0
4/4  Y  1  0/0  0/0  0/0
0/0  Y  1  0/0  2/2  0/0
0/0  Y  1  0/0  3/3  0/0
In this example, the commands move all tables on `dbspace_x` to `dbspace_y`.

```
SELECT 'ALTER TABLE ' || owner || '.' ||
object_name || ' MOVE TO dbspace_y;'  
FROM sp_iqdbspaceobjectinfo()  
WHERE object_type = 'table' AND  
dbspace_name = 'dbspace_x';
```

The following `ALTER TABLE` commands are the result:

```
ALTER TABLE DBA.dt1 MOVE TO dbspace_y;
ALTER TABLE DBA.dt2 MOVE TO dbspace_y;
ALTER TABLE DBA.dt3 MOVE TO dbspace_y;
```

### sp_iqdroplogin Procedure

Drops an SAP Sybase IQ user account.

**Syntax1**

```
call sp_iqdroplogin (‘userid’)  
```

**Syntax2**

```
sp_iqdroplogin ‘userid’  
```

**Syntax3**

```
sp_iqdroplogin userid  
```

**Syntax4**

```
sp_iqdroplogin (‘userid’)  
```

**Privileges**

Requires the MANAGE ANY USER system privilege. Users without MANAGE ANY USER system privilege must be granted EXECUTE permission.

**Usage**

### Table 18. Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>ID of the user to drop.</td>
</tr>
</tbody>
</table>
Description

sp_iqdroplogin drops the specified user.

Examples

These commands all remove the user rose:

sp_iqdroplogin 'rose'
sp_iqdroplogin rose
call sp_iqdroplogin ('rose')

sp_iqemptyfile Procedure

Empties a dbfile and moves the objects in the dbfile to another available read-write dbfile in the same dbspace. Not available for files in an RLV dbspace.

Syntax

sp_iqemptyfile (logical-file--name)

Privileges

Requires at least one system privilege from each group:

<table>
<thead>
<tr>
<th>Group 1</th>
<th>Group 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP DATABASE</td>
<td>INSERT ANY TABLE</td>
</tr>
<tr>
<td>SERVER OPERATOR</td>
<td>UPDATE ANY TABLE</td>
</tr>
<tr>
<td>ALTER DATABASE</td>
<td>DELETE ANY TABLE</td>
</tr>
<tr>
<td></td>
<td>ALTER ANY TABLE</td>
</tr>
<tr>
<td></td>
<td>LOAD ANY TABLE</td>
</tr>
<tr>
<td></td>
<td>TRUNCATE ANY TABLE</td>
</tr>
<tr>
<td></td>
<td>ALTER ANY OBJECT</td>
</tr>
</tbody>
</table>

Users without the required system privileges must be granted EXECUTE permission to run the stored procedure.

Description

sp_iqemptyfile empties a dbfile. The dbspace must be read-only before you can execute the sp_iqemptyfile procedure. The procedure moves the objects in the file to another available read-write dbfile in the same dbspace. If there is no other read-write dbfile available, then SAP Sybase IQ displays an error message.
Note: In a multiplex environment, you can run `sp_iqemptyfile` only on the coordinator. There must be one read-write dbspace available for the procedure to succeed.

If the dbfile is in an RLV dbspace, then this error message displays:

```
Cannot empty files in an rlv store dbspace.
```

Example
Empties dbfile `dbfile1`:
`sp_iqemptyfile 'dbfile1'`

**sp_iqestdbspaces Procedure**
Estimates the number and size of dbspaces needed for a given total index size.

**Syntax**
```
sp_iqestdbspaces ( db_size_in_bytes, iq_page_size,
min_#_of_bytes, max_#_of_bytes )
```

**Privileges**
Requires the MANAGE ANY DBSPACE or ALTER DATABASE system privileges. Users without one of these system privileges must be granted EXECUTE permission.

**Description**
Displays information about the number and size of dbspace segments based on the size of the database, the IQ page size, and the range of bytes per dbspace segment. This procedure assumes that the database was created with the default block size for the specified IQ page size; otherwise, the returned estimated values are incorrect.

**Table 19. sp_iqestdbspaces Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>db_size_in_bytes</code></td>
<td>deci-mal(16)</td>
<td>Size of the database in bytes.</td>
</tr>
<tr>
<td><code>iq_page_size</code></td>
<td>smallint</td>
<td>The page size defined for the IQ segment of the database (must be a power of 2 between 65536 and 524288; the default is 131072).</td>
</tr>
<tr>
<td><code>min_#_of_bytes</code></td>
<td>int</td>
<td>The minimum number of bytes per dbspace segment. The default is 20,000,000 (20MB).</td>
</tr>
<tr>
<td><code>max_#_of_bytes</code></td>
<td>int</td>
<td>The maximum number of bytes per dbspace segment. The default is 2,146,304,000 (2.146GB).</td>
</tr>
</tbody>
</table>

**Usage**
`sp_iqestdbspaces` reports several recommendations, depending on how much of the data is unique:
### Recommendation

<table>
<thead>
<tr>
<th>Recommendation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>min</strong></td>
<td>If there is little variation in data, you can choose to create only the dbspace segments of the sizes recommended as <strong>min</strong>. These recommendations reflect the best possible compression on data with the least possible variation.</td>
</tr>
<tr>
<td><strong>avg</strong></td>
<td>If your data has an average amount of variation, create the dbspace segments recommended as <strong>min</strong>, plus additional segments of the sizes recommended as <strong>avg</strong>.</td>
</tr>
<tr>
<td><strong>max</strong></td>
<td>If your data has a high degree of variation (many unique values), create the dbspace segments recommended as <strong>min</strong>, <strong>avg</strong>, and <strong>max</strong>.</td>
</tr>
<tr>
<td><strong>spare</strong></td>
<td>If you are uncertain about the number of unique values in your data, create the dbspace segments recommended as <strong>min</strong>, <strong>avg</strong>, <strong>max</strong>, and <strong>spare</strong>. You can always delete unused segments after loading your data, but creating too few can cost you some time.</td>
</tr>
</tbody>
</table>

---

**sp_iqfile Procedure**

Displays detailed information about each dbfile in a dbspace.

**Syntax**

```
sp_iqfile [ dbspace-name ]
```

**Applies to**

Simplex and multiplex.

**Privileges**

Requires the MANAGE ANY DBSPACE system privilege. Users without the MANAGE ANY DBSPACE system privilege must be granted EXECUTE permission.

**Description**

**sp_iqfile** displays the usage, properties, and types of data in each dbfile in a dbspace. You can use this information to determine whether data must be moved, and for data that has been moved, whether the old versions have been deallocated.
<table>
<thead>
<tr>
<th><strong>Column Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSpaceName</td>
<td>Name of the dbspace as specified in the CREATE DBSPACE statement. Dbspace names are always case-insensitive, regardless of the CREATE DATABASE...CASE IGNORE or CASE RESPECT specification.</td>
</tr>
<tr>
<td>DBFileName</td>
<td>Logical file name.</td>
</tr>
<tr>
<td>Path</td>
<td>Location of the physical file or raw partition.</td>
</tr>
<tr>
<td>SegmentType</td>
<td>Type of dbspace (MAIN, TEMPORARY, or RLV).</td>
</tr>
<tr>
<td>RWMode</td>
<td>Mode of the dbspace: always read-write (RW).</td>
</tr>
<tr>
<td>Online</td>
<td>T (online) or F (offline).</td>
</tr>
<tr>
<td>Usage</td>
<td>Percent of dbspace currently in use by this file in the dbspace. When run against a secondary node in a multiplex configuration, this column displays NA.</td>
</tr>
<tr>
<td>DBFileSize</td>
<td>Current size of the file or raw partition. For a raw partition, this size value can be less than the physical size.</td>
</tr>
<tr>
<td>Reserve</td>
<td>Reserved space that can be added to this file in the dbspace.</td>
</tr>
<tr>
<td>StripeSize</td>
<td>Always 1, if disk striping is on.</td>
</tr>
<tr>
<td>BlkTypes</td>
<td>Space used by both user data and internal system structures.</td>
</tr>
<tr>
<td>FirstBlk</td>
<td>First IQ block number assigned to the file.</td>
</tr>
<tr>
<td>LastBlk</td>
<td>Last IQ block number assigned to the file.</td>
</tr>
<tr>
<td>OkToDrop</td>
<td>&quot;Y&quot; indicates the file can be dropped; otherwise &quot;N&quot;.*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Identifier</strong></th>
<th><strong>Block Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Active Version</td>
</tr>
<tr>
<td>B</td>
<td>Backup Structures</td>
</tr>
<tr>
<td>C</td>
<td>Checkpoint Log</td>
</tr>
<tr>
<td>D</td>
<td>Database Identity</td>
</tr>
<tr>
<td>F</td>
<td>Free list</td>
</tr>
<tr>
<td>G</td>
<td>Global Free list Manager</td>
</tr>
<tr>
<td>H</td>
<td>Header Blocks of the Free List</td>
</tr>
<tr>
<td>I</td>
<td>Index Advice Storage</td>
</tr>
</tbody>
</table>

---

*Note: This document is a part of the SAP Sybase IQ tutorial and is used for educational purposes.*
*The multiplex commit identity block (actually 128 blocks) exists in all IQ databases, even though it is not used by simplex databases.

**Example**
Displays information about the files in the dbspaces:

```
sp_iqfile;
```

```sql
sp_iqfile;
DBSpaceName,DBFileName,Path,SegmentType,RWMode,Online,
Usage,DBFileSize,Reserve,StripeSize,BlkTypes,FirstBlk,
LastBlk,OkToDrop

'IQ_SYSTEM_MAIN','IQ_SYSTEM_MAIN','/sun1-c1/users/smith/mpx/m/
mpx_db.iq','MAIN','RW','T','21','
2.92G','0B','1K','1H,76768F,32D,19A,1850,128M,34B,32C'
,1,384000,'N'

'mpx_main1','mpx_main1','/sun1-c1/users/smith/mpx/m/
mpx_main1.iq','MAIN','RW','T','1'
',100M','0B','1K','1H',1045440,1058239,'N'

'IQ_SHARED_TEMP','sharedfile1_bcp','/sun1-c1/users/smith/mpx/m/
fl1','SHARED_TEMP','RO','T','0'
,'50M','0B','1K','1H',1,6400,'N'

'IQ_SHARED_TEMP','sharedfile2_bcp','/sun1-c1/users/smith/mpx/m/
f2','SHARED_TEMP','RO','T','0'
,'50M','0B','1K','1H',1045440,1051839,'N'

'IQ_SYSTEM_TEMP','IQ_SYSTEM_TEMP','/sun1-c1/users/smith/mpx/m/
mpx_db.iqtmp','TEMPORARY','RW','
'T','1','2.92G','0B','1K','1H,64F,33A',1,384000,'N'
```
sp_iqmodifyadmin Procedure

Sets an option on a named login policy to a certain value. If no login policy is specified, the option is set on the root policy. In a multiplex, sp_iqmodifyadmin takes an optional parameter that is the multiplex server name.

Syntax 1

call sp_iqmodifyadmin ('policy_option_name', 'value_in', ['login_policy_name'] )

Syntax 2

sp_iqmodifyadmin 'policy_option_name', 'value_in', 'login_policy_name'

Syntax 3

sp_iqmodifyadmin policy_option_name, value_in, ,login_policy_name

Syntax 4

sp_iqmodifyadmin 'policy_option_name', 'value_in', 'login_policy_name', 'server_name'

Usage

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>policy_option_name</td>
<td>The login policy option to be changed.</td>
</tr>
<tr>
<td>value_in</td>
<td>New value for the login policy option.</td>
</tr>
<tr>
<td>login_policy_name</td>
<td>Policy for which the login policy option is to be</td>
</tr>
<tr>
<td></td>
<td>changed.</td>
</tr>
</tbody>
</table>

Permissions

Requires the MANAGE ANY LOGIN POLICY system privilege.

Examples

Sets the login option locked to ON for the policy named lockeduser:

call sp_iqmodifyadmin ('locked', 'on', 'lockeduser')

Sets the login option locked to ON for the policy named lockeduser on the multiplex server named Writer1:

call sp_iqmodifyadmin ('locked', 'on', 'lockeduser', 'Writer1')
**sp_iqmodifylogin Procedure**
Assigns a user to a login policy.

**Syntax1**
call sp_iqmodifylogin 'userid', ['login_policy_name']

**Syntax2**
sp_iqmodifylogin 'userid', ['login_policy_name']

**Privileges**
Requires the MANAGE ANY USER system privilege.

**Usage**

**Table 21. Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>Variable that holds the name of the account to modify.</td>
</tr>
<tr>
<td>login_policy_name</td>
<td>(Optional) Specifies the name of the login policy to which the user will be assigned. If no login policy name is specified, the user is assigned to the root login policy.</td>
</tr>
</tbody>
</table>

**Examples**
Assigns user joe to a login policy named expired_password:
sp_iqmodifylogin 'joe', 'expired_password'

Assigns user joe to the root login policy:
call sp_iqmodifylogin ('joe')

**sp_iqobjectinfo Procedure**
Returns partitions and dbspace assignments of database objects and subobjects.

**Syntax**
sp_iqobjectinfo [ owner_name ] [ , object_name ] [ , object-type ]

**Privileges**
No specific system privilege is required to run this procedure.
### Usage

#### Table 22. Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner_name</td>
<td>Owner of the object. If specified, <code>sp_iqobjectinfo</code> displays output only for tables with the specified owner. If not specified, <code>sp_iqobjectinfo</code> displays information on tables for all users in the database.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the table. If not specified, <code>sp_iqobjectinfo</code> displays information on all tables in the database.</td>
</tr>
<tr>
<td>object-type</td>
<td>Valid table object types. If the object-type is a table, it must be enclosed in quotation marks.</td>
</tr>
</tbody>
</table>

All parameters are optional, and any parameter may be supplied independent of the value of another parameter.

Use input parameters with `sp_iqobjectinfo`; you can query the results of the `sp_iqobjectinfo` and it performs better if you use input parameters rather than using predicates in the `WHERE` clause of the query. For example, Query A is written as:

```sql
SELECT COUNT(*) FROM sp_iqobjectinfo()
WHERE owner = 'DBA'
AND object_name = 'tab_case510'
AND object_type = 'table'
AND sub_object_name is NULL
AND dbspace_name = 'iqmain7'
AND partition_name = 'P1'
```

Query B is Query A rewritten to use `sp_iqobjectinfo` input parameters:

```sql
SELECT COUNT(*) FROM sp_iqobjectinfo('DBA','tab_case510','table')
WHERE sub_object_name is NULL
AND dbspace_name = 'iqmain7'
AND PARTITION_NAME = 'P1'
```

Query B returns results faster than Query A. When the input parameters are passed to `sp_iqobjectinfo`, the procedure compares and joins fewer records in the system tables, thus doing less work compared to Query A. In Query B, the predicates are applied in the procedure itself, which returns a smaller result set, so a smaller number of predicates is applied in the query.

The `sp_iqobjectinfo` stored procedure supports wildcard characters for interpreting `owner_name`, `object_name`, and `object_type`. It shows information for all dbspaces that match the given pattern in the same way the `LIKE` clause matches patterns inside queries.
**Description**
Returns all the partitions and the dbspace assignments of a particular or all database objects (of type table) and its subobjects. The subobjects are columns, indexes, primary key, unique constraints, and foreign keys.

**Table 23. sp_iqobjectinfo columns**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>Name of the owner of the object.</td>
</tr>
<tr>
<td>object_name</td>
<td>Name of the object (of type table) located on the dbspace.</td>
</tr>
<tr>
<td>sub_object_name</td>
<td>Name of the object located on the dbspace.</td>
</tr>
<tr>
<td>object_type</td>
<td>Type of the object (column, index, primary key, unique constraint, foreign key, partition, or table).</td>
</tr>
<tr>
<td>object_id</td>
<td>Global object ID of the object.</td>
</tr>
<tr>
<td>id</td>
<td>Table ID of the object.</td>
</tr>
<tr>
<td>dbspace_name</td>
<td>Name of the dbspace on which the object resides. The string “[multiple]” appears in a special meta row for partitioned objects. The [multiple] row indicates that multiple rows follow in the output to describe the table or column.</td>
</tr>
<tr>
<td>partition_name</td>
<td>Name of the partition for the given object.</td>
</tr>
</tbody>
</table>

**Examples**

**Note:** These examples show objects in the iqdemo database to better illustrate output. iqdemo includes a sample user dbspace named iq_main that may not be present in your own databases.

Displays information about partitions and dbspace assignments of a specific database object and subobjects owned by a specific user:

```
sp_iqobjectinfo GROUPO,Departments
```

<table>
<thead>
<tr>
<th>owner</th>
<th>object_name</th>
<th>sub_object_name</th>
<th>object_type</th>
<th>object_id</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>(NULL)</td>
<td>table</td>
<td>3</td>
<td>632</td>
</tr>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>DepartmentID</td>
<td>column</td>
<td>3</td>
<td>633</td>
</tr>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>DepartmentName</td>
<td>column</td>
<td>3</td>
<td>634</td>
</tr>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>DepartmentHeadID</td>
<td>column</td>
<td>3</td>
<td>635</td>
</tr>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>DepartmentsKey</td>
<td>primary</td>
<td>83</td>
<td>738</td>
</tr>
<tr>
<td>GROUPO</td>
<td>Departments</td>
<td>PK_DepartmentHeadID_EmployeeID</td>
<td>foreign</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Displays information about partitions and dbspace assignments of a specific database object and subobjects owned by a specific user for object-type table:

**sp_iqobjectinfo** `DBA, sale, 'table'

```
<table>
<thead>
<tr>
<th>owner</th>
<th>object_name</th>
<th>sub_object_name</th>
<th>object_type</th>
<th>object_id</th>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBA</td>
<td>sale</td>
<td>(NULL)</td>
<td>table</td>
<td>3698</td>
<td>742</td>
</tr>
<tr>
<td>DBA</td>
<td>sale</td>
<td>prod_id</td>
<td>column</td>
<td>3699</td>
<td>742</td>
</tr>
<tr>
<td>DBA</td>
<td>sale</td>
<td>month_num</td>
<td>column</td>
<td>3700</td>
<td>742</td>
</tr>
<tr>
<td>DBA</td>
<td>sale</td>
<td>rep_id</td>
<td>column</td>
<td>3701</td>
<td>742</td>
</tr>
<tr>
<td>DBA</td>
<td>sale</td>
<td>sales</td>
<td>column</td>
<td>3702</td>
<td>742</td>
</tr>
</tbody>
</table>
```

**sp_iqs spaceused Procedure**

Shows information about space available and space used in the IQ store, IQ temporary store, RLV store, and IQ global and local shared temporary stores.

**Syntax**

```sql
sp_iqs spaceused(out mainKB unsigned bigint,
                  out mainKBUsed unsigned bigint,
                  out tempKB unsigned bigint,
                  out tempKBUsed unsigned bigint,
                  out shTempTotalKB unsigned bigint,
                  out shTempTotalKBUsed unsigned bigint,
                  out shTempLocalKB unsigned bigint,
                  out shTempLocalKBUsed unsigned bigint,
                  out rlvLogKB unsigned bigint,
                  out rlvLogKBUsed unsigned bigint)
```

**Applies to**

Simplex and multiplex.
Privileges
Requires the ALTER DATABASE, MANAGE ANY DBSPACE, or MONITOR system privileges. Users without one of these system privileges must be granted EXECUTE permission.

Usage
sp_iqspaceused returns several values as unsigned bigint out parameters. This system stored procedure can be called by user-defined stored procedures to determine the amount of main, temporary, and RLV store space in use.

Description
sp_iqspaceused returns a subset of the information provided by sp_iqstatus, but allows the user to return the information in SQL variables to be used in calculations.

If run on a multiplex database, this procedure applies to the server on which it runs. Also returns space used on IQ_SHARED_TEMP.

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mainKB</td>
<td>The total IQ main store space, in kilobytes.</td>
</tr>
<tr>
<td>mainKBUsed</td>
<td>The number of kilobytes of IQ main store space used by the database. Secondary multiplex nodes return '(Null)'.</td>
</tr>
<tr>
<td>tempKB</td>
<td>The total IQ temporary store space, in kilobytes.</td>
</tr>
<tr>
<td>tempKBUsed</td>
<td>The number of kilobytes of total IQ temporary store space in use by the database.</td>
</tr>
<tr>
<td>shTempTotalKB</td>
<td>The total IQ global shared temporary store space, in kilobytes.</td>
</tr>
<tr>
<td>shTempLocalKB</td>
<td>The total IQ local shared temporary store space, in kilobytes.</td>
</tr>
<tr>
<td>shTempLocalKBUsed</td>
<td>The number of kilobytes of IQ local shared temporary store space in use by the database.</td>
</tr>
<tr>
<td>rlvLogKB</td>
<td>The total RLV store space, in kilobytes.</td>
</tr>
<tr>
<td>rlvLogKBUsed</td>
<td>The number of kilobytes of RLV store space in use by the database.</td>
</tr>
</tbody>
</table>

Example
sp_iqspaceused requires seven output parameters. Create a user-defined stored procedure myspace that declares the seven output parameters, then calls sp_iqspaceused:
create or replace procedure dbo.myspace()
begin
    declare mt unsigned bigint;
    declare mu unsigned bigint;
    declare tt unsigned bigint;
    declare tu unsigned bigint;
    declare gt unsigned bigint;
    declare gu unsigned bigint;
    declare lt unsigned bigint;
    declare lu unsigned bigint;
    declare rvlt unsigned bigint;
    declare rvlu unsigned bigint;
    call sp_iqspaceused(mt,mu,tt,tu,gt,gu,lt,lu, rvlt, rvlu);
    select cast(mt/1024 as unsigned bigint) as mainMB,
           cast(mu/1024 as unsigned bigint) as mainusedMB,
           mu*100/mt as mainPerCent,
           cast(tt/1024 as unsigned bigint) as tempMB,
           cast(tu/1024 as unsigned bigint) as tempusedMB,
           tu*100/tt as tempPerCent,
           cast(gt/1024 as unsigned bigint) as shTempTotalKB,
           if gu=0 then 0 else gu*100/gt endif as globalshtempPerCent,
           cast(lt/1024 as unsigned bigint) as shTempLocalMB,
           cast(lu/1024 as unsigned bigint) as shTempLocalKBUsed,
           if lt=0 then 0 else lu*100/lt endif as localshtempPerCent,
           cast(rvlt/1024 as unsigned bigint) as rlvLogKB,
           cast(rvlu/1024 as unsigned bigint) as rlvLogKBUsed;
end

To display the output of **sp_iqspaceused**, execute **myspace**:

myspace

**sp_iqsysmon Procedure**

Monitors multiple components of SAP Sybase IQ, including the management of buffer cache, memory, threads, locks, I/O functions, and CPU utilization.

**Batch Mode Syntax**

```
sp_iqsysmon start_monitor
sp_iqsysmon stop_monitor [, “section(s)” ]
```

or

```
sp_iqsysmon “time-period” [, “section(s)” ]
```

**File Mode Syntax**

```
sp_iqsysmon start_monitor, ‘filemode’ [, “monitor-options” ]
sp_iqsysmon stop_monitor
```

**Privileges**

Requires the MONITOR system privilege. Users without the MONITOR system privilege must be granted EXECUTE permission to run the stored procedure.
**Batch Mode Usage**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_monitor</td>
<td>Starts monitoring.</td>
</tr>
<tr>
<td>stop_monitor</td>
<td>Stops monitoring and displays the report.</td>
</tr>
<tr>
<td>time-period</td>
<td>The time period for monitoring, in the form HH:MM:SS.</td>
</tr>
<tr>
<td>section(s)</td>
<td>The abbreviation for one or more sections to be shown by sp_iqsysmon. If you specify more than one section, separate the section abbreviations using spaces, and enclose the list in single or double quotes. The default is to display all sections. For sections related to the IQ store, you can specify main or temporary store by prefixing the section abbreviation with “m” or “t”, respectively. Without the prefix, both stores are monitored. For example, if you specify “mbufman”, only the IQ main store buffer manager is monitored. If you specify “mbufman tbufman” or “bufman”, both the main and temporary store buffer managers are monitored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Report Section or IQ Component</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer manager</td>
<td>(m/t)bufman</td>
</tr>
<tr>
<td>Buffer pool</td>
<td>(m/t)bufpool</td>
</tr>
<tr>
<td>Prefetch management</td>
<td>(m/t)prefetch</td>
</tr>
<tr>
<td>Free list management</td>
<td>(m/t)freelist</td>
</tr>
<tr>
<td>Buffer allocation</td>
<td>(m/t)bufalloc</td>
</tr>
<tr>
<td>Memory management</td>
<td>memory</td>
</tr>
<tr>
<td>Thread management</td>
<td>threads</td>
</tr>
<tr>
<td>CPU utilization</td>
<td>cpu</td>
</tr>
<tr>
<td>Transaction management</td>
<td>txn</td>
</tr>
<tr>
<td>Server context statistics</td>
<td>server</td>
</tr>
<tr>
<td>Report Section or IQ Component</td>
<td>Abbreviation</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Catalog statistics</td>
<td>catalog</td>
</tr>
</tbody>
</table>

**Note:** The SAP Sybase IQ components Disk I/O and Lock Manager are not currently supported by *sp_iqsysmon*.

### File Mode Usage

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_monitor</td>
<td>Starts monitoring.</td>
</tr>
<tr>
<td>stop_monitor</td>
<td>Stops monitoring and writes the remaining output to the log file.</td>
</tr>
<tr>
<td>filemode</td>
<td>Specifies that <em>sp_iqsysmon</em> is running in file mode. In file mode, a sample of statistics appear for every interval in the monitoring period. By default, the output is written to a log file named <code>dbname.connid-iqmon</code>. Use the <em>file_suffix</em> option to change the suffix of the output file. See the monitor_options parameter for a description of the file_suffix option.</td>
</tr>
<tr>
<td>monitor_options</td>
<td>The monitor_options string</td>
</tr>
</tbody>
</table>

The *monitor_options* string can include one or more options:
<table>
<thead>
<tr>
<th>monitor_options String Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>-interval</strong> <em>seconds</em></td>
<td>Specifies the reporting interval, in seconds. A sample of monitor statistics is output to the log file after every interval. The default is every 60 seconds, if the <em>-interval</em> option is not specified. The minimum reporting interval is 2 seconds. If the interval specified for this option is invalid or less than 2 seconds, the interval is set to 2 seconds. The first display shows the counters from the start of the server. Subsequent displays show the difference from the previous display. You can usually obtain useful results by running the monitor at the default interval of 60 seconds during a query with performance problems or during a time of day that generally has performance problems. A very short interval may not provide meaningful results. The interval should be proportional to the job time; 60 seconds is usually more than enough time.</td>
</tr>
<tr>
<td><strong>-file_suffix</strong> <em>suffix</em></td>
<td>Creates a monitor output file named dbname.connid-suffix. If you do not specify the <em>-file_suffix</em> option, the suffix defaults to iqmon. If you specify the <em>-file_suffix</em> option and do not provide a suffix or provide a blank string as a suffix, no suffix is used.</td>
</tr>
<tr>
<td><strong>-append</strong> or <strong>-truncate</strong></td>
<td>Directs <em>sp_iqsysmon</em> to append to the existing output file or truncate the existing output file, respectively. Truncate is the default. If both options are specified, the option specified later in the string takes precedence.</td>
</tr>
</tbody>
</table>
### Description

The **sp_iqsysmon** stored procedure monitors multiple components of SAP Sybase IQ, including the management of buffer cache, memory, threads, locks, I/O functions, and CPU utilization.

The **sp_iqsysmon** procedure supports two modes of monitoring:

- **Batch mode** –
  
  In batch mode, **sp_iqsysmon** collects the monitor statistics for the period between starting and stopping the monitor or for the time period specified in the `time-period` parameter. At the end of the monitoring period, **sp_iqsysmon** displays a list of consolidated statistics.

### Usage Syntax Examples

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sp_iqsysmon start_monitor</code></td>
<td>Starts the monitor in batch mode and displays all sections for the main and temporary stores</td>
</tr>
<tr>
<td><code>sp_iqsysmon stop_monitor</code></td>
<td></td>
</tr>
<tr>
<td><code>sp_iqsysmon start_monitor</code></td>
<td>Starts the monitor in batch mode and displays the Buffer Manager and Buffer Pool statistics for the main store</td>
</tr>
<tr>
<td><code>sp_iqsysmon stop_monitor “mbufman mbufpool”</code></td>
<td></td>
</tr>
<tr>
<td><code>sp_iqsysmon “00:00:10”, “mbufpool memory”</code></td>
<td>Runs the monitor in batch mode for 10 seconds and displays the consolidated statistics at the end of the time period</td>
</tr>
<tr>
<td><code>sp_iqsysmon start_monitor, ‘filemode’, “-interval 5 -sections mbufpool memory”</code></td>
<td>Starts the monitor in file mode and writes statistics for Main Buffer Pool and Memory Manager to the log file every 5 seconds</td>
</tr>
<tr>
<td><code>sp_iqsysmon stop_monitor</code></td>
<td></td>
</tr>
</tbody>
</table>

### Monitor Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-section section(s)</code></td>
<td>Specifies the abbreviation of one or more sections to write to the monitor log file. The default is to write all sections. The abbreviations specified in the sections list in file mode are the same abbreviations used in batch mode. When more than one section is specified, spaces must separate the section abbreviations. If the <code>-section</code> option is specified with no sections, none of the sections are monitored. An invalid section abbreviation is ignored and a warning is written to the IQ message file.</td>
</tr>
</tbody>
</table>
**sp_iqsysmon** in batch mode is similar to the SAP® Sybase Adaptive Server Enterprise procedure **sp_sysmon**.

- **File mode** –
  In file mode, **sp_iqsysmon** writes the sample statistics in a log file for every interval period between starting and stopping the monitor.
  The first display in file mode shows the counters from the start of the server. Subsequent displays show the difference from the previous display.
  **sp_iqsysmon** in file mode is similar to the IQ UTILITIES command **START MONITOR** and **STOP MONITOR** interface.

**Batch Mode Examples**
Prints monitor information after 10 minutes:

```
sp_iqsysmon "00:10:00"
```

Prints only the Memory Manager section of the **sp_iqsysmon** report after 5 minutes:

```
sp_iqsysmon "00:05:00", memory
```

Starts the monitor, executes two procedures and a query, stops the monitor, then prints only the Buffer Manager section of the report:

```
sp_iqsysmon start_monitor
go
execute proc1
go
execute proc2
go
select sum(total_sales) from titles
go
sp_iqsysmon stop_monitor, bufman
go
```

Prints only the Main Buffer Manager and Main Buffer Pool sections of the report after 20 minutes:

```
sp_iqsysmon "00:02:00", "mbufman mbufpool"
```

**File Mode Examples**
Truncates and writes information to the log file every 2 seconds between starting the monitor and stopping the monitor:

```
sp_iqsysmon start_monitor, 'filemode', '-interval 2'
```

```
sp_iqsysmon stop_monitor
```

Appends output for only the Main Buffer Manager and Memory Manager sections to an ASCII file with the name *dbname.connid-testmon*. For the database *iqdemo*, writes results in the file *iqdemo.2-testmon*:

```
sp_iqsysmon start_monitor, 'filemode',
"-file_suffix testmon -append -section mbufman memory"
```
sp_iqsysmon stop_monitor

**Example**
Run the monitor in batch mode for 10 seconds and display the consolidated statistics at the end of the time period

```
sp_iqsysmon "00:00:10", "mbufpool memory"
```

```
 Buffer Pool (Main)

STATS-
NAME TOTAL NONE BTREEV BTREEF BV VDO DBEXT DBID SORT
MovedToMRU 0 0 0 0 0 0 0 0 0
MovedToWash 0 0 0 0 0 0 0 0 0
RemovedFromLRU 0 0 0 0 0 0 0 0 0
RemovedFromWash 0 0 0 0 0 0 0 0 0
RemovedInScanMode 0 0 0 0 0 0 0 0 0

STORE GARRAY BARRAY BLKMAP HASH CKPT BM TEST CMID RIDCA LOB
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

STATS-NAME VALUE
Pages 127 (100.0 %)
InUse 4 (3.1 %)
Dirty 1 (0.8 %)
Pinned 0 (0.0 %)
Flushes 0
FlushedBufferCount 0
GetPageFrame 0
GetPageFrameFailure 0
GotEmptyFrame 0
Washed 0
TimesSweepersWoken 0

washTeamSize 0
WashMaxSize 26 (20.5 %)
washNBuffers 4 (3.1 %)
washNDirtyBuffers 1 (0.8 %)
washSignalThreshold 3 (2.4 %)
washNActiveSweepers 0
washIntensity 1

Memory Manager

STATS-NAME VALUE
**sp_iqpassword Procedure**
Changes a user’s password.

*Syntax1*
```
call sp_iqpassword (‘caller_password’, ‘new_password’ [, ‘user_name’])
```

*Syntax2*
```
sp_iqpassword ‘caller_password’, ‘new_password’ [, ‘user_name’]
```

*Privileges*
- None to set your own password.
- The CHANGE PASSWORD system privilege is required to set other users’ passwords.

*Usage*

<table>
<thead>
<tr>
<th>Table 25. Parameters</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>caller_password</td>
<td>Your password. When you are changing your own password, this is your old password. When a user with the CHANGE PASSWORD system privilege is changing another user’s password, caller_password is the of the user making the change.</td>
</tr>
<tr>
<td>new_password</td>
<td>New password for the user, or for loginname.</td>
</tr>
<tr>
<td>user_name</td>
<td>Login name of the user whose password is being changed by by another user with CHANGE PASSWORD system privilege. Do not specify user_name when changing your own password.</td>
</tr>
</tbody>
</table>

*Description*
A user password is an identifier. Any user can change his or her own password using `sp_iqpassword`. The CHANGE PASSWORD system privilege is required to change the password of any existing user.

Identifiers have a maximum length of 128 bytes. They must be enclosed in double quotes or square brackets if any of these conditions are true:
The identifier contains spaces.
The first character of the identifier is not an alphabetic character (as defined below).
The identifier contains a reserved word.
The identifier contains characters other than alphabetic characters and digits.

Alphabetic characters include the alphabet, as well as the underscore character (_), at sign (@), number sign (#), and dollar sign ($). The database collation sequence dictates which characters are considered alphabetic or digit characters.

Examples
Changes the password of the logged-in user from irk103 to exP984:

```sql
sp_iqpassword 'irk103', 'exP984'
```

If the logged-in user has the CHANGE PASSWORD system privilege or joe, the password of user joe from eprr45 to pdi032:

```sql
call sp_iqpassword ('eprr45', 'pdi932', 'joe')
```

**sp_objectpermission System Procedure**
Generates a report on object permissions granted to the specified role, or user name, or the object permissions granted on the specified object or dbspace.

**Syntax**

```sql
dbo.sp_objectpermission ( [object_name], [object_owner], [object_type] )
```

**Arguments**

<table>
<thead>
<tr>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object_name</td>
<td>The name of an object or dbspace or a user or a role. If not specified, object permissions of the current user are reported. Default value is NULL.</td>
</tr>
<tr>
<td>object_owner</td>
<td>The name of the object owner for the specified object name. The object permissions of the specified object owned by the specified object owner are displayed. This parameter must be specified to obtain the object permissions of an object owned by another user or role. Default value is NULL.</td>
</tr>
</tbody>
</table>
### Arguments

<table>
<thead>
<tr>
<th>object_type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Valid values are:</td>
</tr>
<tr>
<td></td>
<td>• TABLE*</td>
</tr>
<tr>
<td></td>
<td>• VIEW</td>
</tr>
<tr>
<td></td>
<td>• MATERIALIZED VIEW</td>
</tr>
<tr>
<td></td>
<td>• SEQUENCE</td>
</tr>
<tr>
<td></td>
<td>• PROCEDURE</td>
</tr>
<tr>
<td></td>
<td>• FUNCTION</td>
</tr>
<tr>
<td></td>
<td>• DBSPACE</td>
</tr>
<tr>
<td></td>
<td>• USER</td>
</tr>
</tbody>
</table>

**Note:** * Column-level object permissions also appear.

If no value is specified, permissions on all object types are returned. Default value is NULL.

### Result Set

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>grantor</td>
<td>char(128)</td>
<td>The user ID of the grantor</td>
</tr>
<tr>
<td>grantee</td>
<td>char(128)</td>
<td>The user ID of the grantee</td>
</tr>
<tr>
<td>object_name</td>
<td>char(128)</td>
<td>The name of the object</td>
</tr>
<tr>
<td>object_type</td>
<td>char(20)</td>
<td>The type of object</td>
</tr>
<tr>
<td>column_name</td>
<td>char(128)</td>
<td>The name of the column</td>
</tr>
<tr>
<td>permission</td>
<td>char(20)</td>
<td>The name of the permission</td>
</tr>
<tr>
<td>grantable</td>
<td>char(1)</td>
<td>Whether or not the permission is grantable</td>
</tr>
</tbody>
</table>

### Remarks

All arguments are optional and can generate these reports:

- If input is an object (table, view, procedure, function, sequence, and so on), procedure displays list of all roles and user that have different object permission on the object.
- If input is a role or user, procedure displays list of all object privileges granted to the role or input. When executing `sp_objectpermission` to display object permissions of a user or a role, the object permissions that are inherited through role grants also.
- If input is a dbspace name, procedure displays list of all user or roles that have CREATE permission on the specified dbspace.
By default, object type is NULL and the object permissions for all existing object types matching the specified object name appear.

**Privileges**

- Any user can execute `sp_objectpermission` to obtain all the object permissions granted to him- or herself,
- Object owners can execute `sp_objectpermission` to obtain the object permissions for self-owned objects.
- `MANAGE ANY OBJECT PRIVILEGE` system privilege is required to obtain object permissions that are granted:
  - On objects owned by other users
  - To other users
- `MANAGE ANY OBJECT PRIVILEGE` system privilege or role administrator is required to obtain object permissions that are granted:
  - On objects owned by a role
  - To a role.
- `MANAGE ANY DBSPACE` system privilege required to obtain permissions of a dbspace.

**Example**

The following GRANT statements are executed:

```
GRANT SERVER OPERATOR TO r4;
GRANT BACKUP DATABASE TO r3 WITH ADMIN OPTION;
GRANT DROP CONNECTION TO r3 WITH ADMIN ONLY OPTION;
GRANT MONITOR TO r2;GRANT CHECKPOINT TO r1;
GRANT ROLE r2 TO r1 WITH ADMIN OPTION;
GRANT ROLE r3 TO r2 WITH NO ADMIN OPTION;
GRANT ROLE r4 TO r3 WITH ADMIN ONLY OPTION;
```

Consider these object permissions:

- `r5` owns a table named `test_tab` and a procedure named `test_proc` in the database.
- `u5`, which has administrative rights over `r5`, grants the following permissions:
  - GRANT SELECT ON `r5.test_tab` TO `r2` WITH GRANT OPTION;
  - GRANT SELECT (c1), UPDATE (c1) ON `r5.test_tab` TO `r6` WITH GRANT OPTION;
  - GRANT EXECUTE ON `r5.test_proc` TO `r3`;
- `u6`, which has administrative rights over `r6`, grants the following permissions:
  - GRANT SELECT (c1), REFERENCES (c1) ON `r5.test_tab` TO `r3`;

If `sp_objectpermission( 'r1' )` is executed, output is similar to:
If `sp_objectpermission( 'test_tab', 'r5', 'table' )` is executed, output is similar to:

<table>
<thead>
<tr>
<th>granter</th>
<th>grantees</th>
<th>object_name</th>
<th>owner</th>
<th>object_type</th>
<th>column_name</th>
<th>permission</th>
<th>grantable</th>
</tr>
</thead>
<tbody>
<tr>
<td>u5</td>
<td>r2</td>
<td>test_tab</td>
<td>r5</td>
<td>TABLE</td>
<td>NULL</td>
<td>SELECT</td>
<td>Y</td>
</tr>
<tr>
<td>u6</td>
<td>r3</td>
<td>test_tab</td>
<td>r5</td>
<td>COLUMN</td>
<td>c1</td>
<td>SELECT</td>
<td>N</td>
</tr>
<tr>
<td>u6</td>
<td>r3</td>
<td>test_tab</td>
<td>r5</td>
<td>COLUMN</td>
<td>c1</td>
<td>REFERENCES</td>
<td>N</td>
</tr>
<tr>
<td>u6</td>
<td>r3</td>
<td>test_proc</td>
<td>r5</td>
<td>PROCEDURE</td>
<td>NULL</td>
<td>EXECUTE</td>
<td>N</td>
</tr>
</tbody>
</table>

**sp_sys_priv_role_info System Procedure**

Generates a report to map a system privilege to the corresponding system role. A single row is returned for each system privilege.

**Syntax**

`sp_sys_priv_role_info()`
### Result Set

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sys_priv_name</td>
<td>char(128)</td>
<td>The name of the system privilege</td>
</tr>
<tr>
<td>sys_priv_role_name</td>
<td>char(128)</td>
<td>The role name corresponding to the system privilege.</td>
</tr>
<tr>
<td>sys_priv_id</td>
<td>unsigned int</td>
<td>The id of the system privilege</td>
</tr>
</tbody>
</table>

### Privileges

none

### `sp_alter_secure_feature_key` System Procedure

Alters a previously-defined secure feature key by modifying the authorization key and/or the feature list.

#### Syntax

```sql
sp_alter_secure_feature_key (name, auth_key, features)
```

#### Arguments

- **name** – the VARCHAR (128) name for the secure feature key you want to alter. A key with the given name must already exist.
- **auth_key** – the CHAR (128) authorization key for the secure feature key. The authorization key must be either a non-empty string of at least six characters, or NULL, indicating that the existing authorization key is not to be changed.
- **features** – the LONG VARCHAR, comma-separated list of secure features that the key can enable. The feature_list can be NULL, indicating that the existing feature_list is not to be changed.

#### Remarks

This procedure allows you to alter the authorization key or feature list of an existing secure feature key.

### Privileges

To use this procedure, you must be the database server owner and have the manage_keys feature enabled on the connection.
sp_create_secure_feature_key System Procedure

Creates a new secure feature key.

Syntax

```
sp_create_secure_feature_key (  
  name,  
  auth_key,  
  features )
```

Arguments

- name – the VARCHAR (128) name for the new secure feature key. This argument cannot be NULL or an empty string.
- auth_key – the CHAR (128) authorization key for the secure feature key. The authorization key must be a non-empty string of at least six characters.
- features – the LONG VARCHAR comma-separated list of secure features that the new key can enable. Specifying "-" before a feature means that the feature is not re-enabled when the secure feature key is set.

Remarks

This procedure creates a new secure feature key that can be given to any user. The system secure feature key is created using the -sk database server option.

Privileges

To use this procedure, you must be the database server owner and have the manage_keys feature enabled on the connection.

sp_drop_secure_feature_key System Procedure

Deletes a secure feature key.

Syntax

```
sp_drop_secure_feature_key ( name )
```

Arguments

- name – the VARCHAR (128) name of the secure feature key to drop.

Remarks

If the named key does not exist, an error is returned. If the named key exists, it is deleted as long as it is not the last secure feature key that is allowed to manage secure features and secure feature keys. For example, the system secure feature key cannot be dropped until there is another key that has the manage_features and manage_keys secure features enabled.
Privileges
To use this procedure, you must be the database server owner and have the manage_keys feature enabled on the connection.

sp_list_secure_feature_keys System Procedure
Returns information about the contents of a directory.

Syntax
sp_list_secure_feature_keys ( )

Result Set

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>VARCHAR(128)</td>
<td>The name of the secure feature key.</td>
</tr>
<tr>
<td>features</td>
<td>LONG VARCHAR</td>
<td>The secure features enabled by the secure feature key.</td>
</tr>
</tbody>
</table>

Remarks
This procedure returns the names of existing secure feature keys, as well as the set of secure features that can be enabled by each key.

If the user has the manage_features and manage_keys secure features enabled, then the procedure returns a list of all secure feature keys.

If the user only has the manage_keys secure feature enabled, then the procedure returns keys that have the same features or a subset of the same features that the current user has enabled.

Privileges
To use this procedure, you must be the database server owner and have the manage_keys feature enabled on the connection.

sp_use_secure_feature_key System Procedure
Enables an existing secure feature key.

Syntax
sp_use_secure_feature_key ( name, sfkey )

Arguments
• name – the VARCHAR (128) name of the secure feature key to be enabled.
• **sfkey** – the CHAR (128) authorization key for the secure feature key being enabled. The authorization key must be at least six characters.

**Remarks**
This procedure enables the secure features that are turned on by the specified secure feature key.

**Privileges**
None.
Appendix: Startup and Connection Parameters

Reference material for startup options and connection parameters for the `start_iq` utility.

**-ec iqsrv16 database server option**

Uses transport-layer security or simple encryption to encrypt all command sequence communication protocol packets (such as DBLib and ODBC) transmitted to and from all clients. TDS packets aren't encrypted.

**Syntax**

```
iqsrv16 -ec encryption-options ...
```

```
encryption-options :
{ NONE | SIMPLE | TLS ( [ FIPS={ Y | N }; ]
   IDENTITY=server-identity-filename;
   IDENTITY_PASSWORD=password ) }, ...
```

**Allowed values**

- **NONE** – accepts connections that aren't encrypted.
- **SIMPLE** – accepts connections that are encrypted with simple encryption. This type of encryption is supported on all platforms, and on previous versions of the database server and clients. Simple encryption doesn't provide server authentication, RSA encryption, or other features of transport-layer security.
- **TLS** – accepts connections that are encrypted with RSA encryption. The TLS parameter accepts the following arguments:
  - **FIPS** – For FIPS-certified RSA encryption, specify `FIPS=Y`. RSA FIPS-certified encryption uses a separate certified library, but is compatible with version 9.0.2 or later clients specifying RSA.

    For a list of FIPS-certified components, see [http://www.sybase.com/detail?id=1061806](http://www.sybase.com/detail?id=1061806).

    The algorithm must match the encryption used to create your certificates.
  - **server-identity-filename** – is the path and file name of the server identity certificate. If you are using FIPS-certified RSA encryption, you must generate your certificates using the RSA algorithm.
• *password* – is the password for the server private key. You specify this password when you create the server certificate.

**Applies to**
NONE and SIMPLE apply to all servers and operating systems.

TLS applies to all servers and operating systems.


**Remarks**
You can use this option to secure communication packets between client applications and the database server using transport-layer security.

The `-ec` option instructs the database server to accept only connections that are encrypted using one of the specified types. You must specify at least one of the supported parameters in a comma-separated list. Connections over the TDS protocol, which include Java applications using jConnect, are always accepted and are never encrypted, regardless of the usage of the `-ec` option. Setting the TDS protocol option to NO disallows these unencrypted TDS connections.

By default, communication packets aren't encrypted, which poses a potential security risk. If you are concerned about the security of network packets, use the `-ec` option. Encryption affects performance only marginally.

If the database server accepts simple encryption, but does not accept unencrypted connections, then any non-TDS connection attempts using no encryption automatically use simple encryption.

Starting the database server with `-ec SIMPLE` tells the database server to only accept connections using simple encryption. TLS connections (RSA and RSA FIPS-certified encryption) fail, and connections requesting no encryption use simple encryption.

If you want the database server to accept encrypted connections over TCP/IP, but also want to be able to connect to the database from the local computer over shared memory, you can specify the `-es` option with the `-ec` option when starting the database server.

The `dbrsa16.dll` file contains the RSA code used for encryption and decryption. The file `dbfips16.dll` contains the code for the FIPS-certified RSA algorithm. When you connect to the database server, if the appropriate file cannot be found, or if an error occurs, a message appears in the database server messages window. The server doesn't start if the specified types of encryption cannot be initiated.

The client's and the server's encryption settings must match or the connection fails except in the following cases:

• If `-ec SIMPLE` is specified on the database server, but `-ec NONE` is not, then connections that do not request encryption can connect and automatically use simple encryption.
If the database server specifies RSA and the client specifies FIPS-certified encryption, or vice versa, the connection succeeds. In these cases, the Encryption connection property returns the value specified by the database server.

**Note:** Separately licensed component required.

FIPS-certified encryption requires a separate license. All strong encryption technologies are subject to export regulations.

**Example**

The following example specifies that connections with no encryption and simple encryption are allowed.

```
iqsrv16 -ec NONE,SIMPLE -x tcpip c:\mydemo.db
```

The following example starts a database server that uses the RSA server certificate rsaserver.id.

```
iqsrv16 -ec TLS(IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test) -x tcpip c:\mydemo.db
```

The following example starts a database server that uses the FIPS-approved RSA server certificate rsaserver.id.

```
iqsrv16 -ec TLS(FIPS=Y;IDENTITY=rsaserver.id;IDENTITY_PASSWORD=test) -x tcpip c:\mydemo.db
```

### -es iqsrv16 database server option

Allows unencrypted connections over shared memory.

**Syntax**

```
iqsrv16 -ec encryption-options -es ...
```

**Applies to**

All servers and operating systems.

**Remarks**

This option is only effective when specified with the -ec option. The -es option instructs the database server to allow unencrypted connections over shared memory. Connections over TCP/IP must use an encryption type specified by the -ec option. This option is useful in situations where you want remote clients to use the -ec option. This option is useful in situations where you want remote clients to use encrypted connections, but for performance reasons you also want to access the database from the local computer with an unencrypted connection.

**Example**

The following example specifies that connections with simple encryption and unencrypted connections over shared memory are allowed.
TDS Communication Parameter

Controls whether the server allows TDS connections.

Usage
TCP/IP, NamedPipes (server side only)

Values
YES, NO

Default
YES

Description
To disallow TDS connections to a database server, set TDS to NO. To ensure that only encrypted connections are made to your server, these port options are the only way to disallow TDS connections.

Example
The following command starts a database server that uses the TCP/IP protocol, but disallows connections from Open Client or jConnect applications.

start_iq -x tcpip(TDS=NO) ...
Index

A

Advanced Security option for Sybase IQ 191
Advanced Security Option 141
AES
definition 192
AES_DECRYPT function
SQL syntax 198
AES_ENCRYPT function
SQL syntax 195
ALTER LDAP SERVER statement 231
ALTER LOGIN POLICY statement syntax 233
ALTER privilege, tables and views grant 86
ALTER ROLE statement 240
ALTER USER statement 242
ASE_BINARY_DISPLAY
ciphertext integrity 219
database option 219

B

backup operations summary 326
binary data
controlling implicit conversion 220
blanks trimming trailing 209, 211
buffer cache
monitoring with sp_iqsysmon 354
bulk load 200

c

case sensitivity
passwords 122
user IDs 122
catalog store
monitoring with 354
change password
dual control option 106
granting 260
revoking 278
change password - single user 105
change password - two users 107
change password dual control enable 106
CHANGE PASSWORD system privilege grant 101
revoke 104
character sets
client file bulk load 208
ciphertext 192
accidental truncation 219
AES_ENCRYPT 194
effect of data types 193, 194
integrity preservation 219
prevent implicit conversion 220
string comparisons 218
client file bulk load character sets 208
errors 208
rollback 208
collations
client file bulk load 208
column encryption 192
communication parameters TDS 374
comparisons
encrypted text 218
compatibility role
revoke 36
compatibility roles 28
SYS_AUTH_DBA_ROLE 32
SYS_AUTH_SA_ROLE 28
SYS_AUTH_SSO_ROLE 31
connect permission 123
CONNECT privilege
GRANT statement 262
CONNECT statement
revoke 279
ConnectFailed event handler 132
connections establishing 233
logical servers 239
managing 131
maximum 128

Administration: User Management and Security 375
CONVERSION_MODE
  ciphertext protection 220
database option 220
CONVERSION_MODE option 220
CREATE LDAP SERVER statement 245
CREATE LOGIN POLICY statement
  syntax 248
CREATE ON statement
  revoke 280
CREATE privilege 94
CREATE privilege, dbspace
  grant 91
CREATE ROLE statement 254
CREATE statement
  grant 264
CREATE USER statement 255
cursors
  connection limit 134

data type conversion
  CONVERSION_MODE option 220
data types
  encrypted columns support 193
  original type preservation 193, 194
database object privileges 84
database options
  ASE_BINARY_DISPLAY 219
  CONVERSION_MODE 220
  for column decryption 219
  for column encryption 219
  maximum string length 292
  STRING_RTRUNCATION 219
database privileges
  inheritance 85
databases
  creating with utility database 158
  loading data into 200
  permission to create and drop 159
  privileges 94
dba password
  change 121
dba user
  unable to manage role 21
DBA user 120
dbo user ID
  views owned by 138
dbspace
  grant CREATE privilege 91
decryption
  AES_DECRYPT function 198
  definition 192
DELETE privilege, tables and views
  grant 86
DROP LDAP SERVER statement 257
DROP LOGIN POLICY statement
  syntax 258
drop role 6, 125
DROP ROLE statement 258
DROP USER statement 260
DROP VIEW statement
  restriction 138
dropping
  users 280
  views 138

e
eigen
  AES_ENCRYPT function 195
  column 141, 192
  communications 374
  data type support 193
  database 141
  definition 192
  definitions of terms 192
  FIPS 141, 191
  RSA 141, 191
  string comparisons 218
event handlers
  ConnectFailed 132
example
  AES_DECRYPT 199, 221
  AES_ENCRYPT 194, 221
  LOAD TABLE ENCRYPTED 200
EXECUTE privilege, procedure, user-defined function
  grant 92
EXECUTE statement
  grant 265
  revoke 281
external authentication
  kerberos 163
  LDAP 163
FIPS 191
  encryption algorithm 192
  support in Sybase IQ 191
FIPS support 141
  functions
    REPLACE function 196
  functions, string
    AES_DECRYPT function 198
    AES_ENCRYPT function 195

G
  global role administrator 10
    adding when creating role 13
    grant to user 15
  global role administrators
    adding 14
    removing 19
  GRANT CHANGE PASSWORD statement 260
  GRANT object-level privileges 86, 265
  GRANT ROLE statement 267
  GRANT SET USER statement 272
  GRANT statement
    CONNECT privilege 262
    new users 123
    passwords 124
  GRANT system privilege statement 274

H
  HEADER SKIP option
    LOAD TABLE statement 213

I
  impersonation 108
    criteria requirement 110
    start 115
    stop 117
    verify current status 117
  INSERT privilege, tables and views
    grant 87
  IPv6 support 152
  IQ_SYSTEM_MAIN
    CREATE privilege 94
  IQ_SYSTEM_TEMP
    CREATE privilege 94
    ISYSDUMMY table
      privileges 83
    ISYSGROUP table
      privileges 83
    ISYSPROCPerm table
      privileges 83
    ISYSTABLEPERM table
      privileges 83
    ISYSUSERPERM table
      privileges 83

K
  kerberos
    licensing requirements 190, 230
  Kerberos authentication 141, 229
  key
    definition 192

L
  LDAP login policy options 237, 252
  LDAP server
    editing object attributes 171
    refresh 173
    suspending 174
  LDAP server configuration object
    altering 231
    activate 171
    create 167
    creating 245
    current status 180
    definition 163
    deleting 174
    dropping 257
    sa_get_ldapserver_status 180
    states 175
    TLS 166
    URL 176
    user authentication 163, 164, 166
    validate 169
    validating 294
  LDAP user authentication 163
    allow standard authentication 165
    current user status 180
    failover 164
    LDAP server configuration object 164
    LDAPUA 165
Licensing 163, 230
login method 165
login policy options 177
login_mode 165
manage users and passwords 179
sa_get_user_status 180
workflow 164
licensing
kerberos 190, 230
LOAD privilege, tables
grant 87
LOAD TABLE
ENCRIPTED clause 199
ENCRIPTED clause example 200
LOAD TABLE statement
FROM clause deprecated 208
HEADER SKIP option 213
new syntax 211
ON PARTIAL INPUT ROW option 214
performance 211
QUOTES option 209
STRIP keyword 211
syntax 200
syntax changes 211
USING keyword 208
WORD SKIP option 214
lockout
automatic 132
logical servers
connections 239
login attempts
exceeding limit 126
login failures 132
login management
list of procedures 133
sp_expireallpasswords 320
sp_iqaddlogin 323
sp_iqcopyloginpolicy 331, 348
login policies 128
altering 233
assigning user to 349
changing 239, 253
copying 331, 348
creating 248
dropping 258
option for locking 125
resetting 126
login policy
assign 130, 131, 179
create 129, 178
delete 130
modify 129, 178
options 235, 249
login policy, root
modify 128, 177
LOGIN_MODE option 297
logins
limiting 131
manage password 101
manage roles
role administrators 21
max_days_since_login
exceeding 126
max_failed_login_attempts
exceeding 126
memory
connection limit 134
monitoring with sp_iqsysmon 354
MIN_PASSWORD_LENGTH option 302
MIN_ROLE_ADMIN option 298
monitor
sp_iqsysmon procedure 354
MPXServerName column 328
multiplex
system procedures 327
named pipes 216
object privileges granted
sp_objectpermission 96
object-level privilege
revoke administrative rights 92
revoke privilege 92
option value
truncation 292
options
ASE_BINARY_DISPLAY 219
CONVERSION_MODE 220
for column decryption 219
for column encryption 219
login policies 239, 253
setting 135, 290
STRING_RTRUNCATION 219
owners
about 84

P

password security 122
passwords
  adding or modifying 361
case sensitivity 122
changing 124, 262
expiration 128
expiring 320
lost 133
minimum length 124, 302
rules 124
setting expiration 131
utility database 158
verifying 124
performance
  monitoring 354
  sp_iqsysmon procedure 354
permissions
  connect 123
  CONNECT privilege 262
  granting passwords 123
  passwords 124
plaintext 192
prefetching
  monitoring with sp_iqsysmon 354
privilege
  inheriting 90
privileges 40
  command-line switches 94
  conflicts 96
dbspace management 94
inheriting 2
INSERT and DELETE, on views 138
listing 83
procedure 95
revoke 95
roles 2
the right to grant 90
WITH GRANT OPTION 90
privileges versus permissions 40
privileges, grant
  ALTER 265
  DELETE 265
  INSERT 265
  LOAD 265
  REFERENCES 265
  SELECT 265
  TRUNCATE 265
  UPDATE 265
privileges, revoke
  ALTER 281
  DELETE 281
  INSERT 281
  LOAD 281
  REFERENCES 281
  SELECT 281
  TRUNCATE 281
  UPDATE 281
procedure, user-defined function
  grant EXECUTE privilege 92
procedures
  owner 84
  security 136
  sp_droplogin 279
  sp_iqdroplogin 279

R

raw devices
  utility database 158
recovery account 133
REFERENCES privilege, tables and views
  grant 88
REPLACE function 196
  in SELECT INTO statement 196
resetting login policies 126
REVOKE CHANGE PASSWORD statement 278
REVOKE database object privilege statement 281
REVOKE object-level privileges 86
REVOKE ROLE statement 282
REVOKE SET USER statement 284
REVOKE system privilege statement 286
Rijndael 192
role
  creating 254
dropping 258
  granting 267
  revoking 282
role access
  procedures 140
role administrator 10
  adding when creating role 12
role administrators
  adding 14
global role administrators 19
minimum number 20, 21
removing 18
replacing existing 15
role-based access control
implementing 2
RBAC 2
workflow 2
role-based security model
implementing 2
RBAC 2
workflow 2
roles
altering 240
managing 2
roles and system privileges granted
sp_has_role 39
roles based access control 1
roles granted
sp_displayroles 38
RSA support 141, 191

S
sa_get_ldapserver_status system procedure 313
scalar value subqueries 137
Secure LDAP
TLS 176
security
Advanced Security Option 141
column encryption 141
database encryption 141
FIPS support 141, 191
IPv6 support 152
Kerberos authentication 141, 229
login failures 132
minimum password length 302
procedures 136
RSA support 141, 191
Sybase IQ Advanced Security option 191
views 136
security by views 136
security management 1
security model 98
SELECT INTO
using REPLACE function 196
SELECT privilege, tables and views
grant 88
SELECT statement
restrictions for view creation 137
sequence generator
grant USAGE privilege 92
SET OPTION statement
syntax 290
SET TEMPORARY OPTION statement
syntax 290
set user
granting 272
revoking 284
SET USER system privilege
grant 113
revoke 118
SETUSER statement
impersonate 292
sp_displayroles system procedure 317
sp_expireallpasswords system procedure 320
sp_has_role function 320
sp_iqaddlogin system procedure 323
sp_iqbackupdetails stored procedure 324
sp_iqbackupsummary stored procedure 326
sp_iqconnection system procedure 327
sp_iqcopypasswordpolicy system procedure 331, 348
sp_iqdbspace system procedure 332
sp_iqdbspaceinfo system procedure 334
sp_iqdbspaceobjectinfo system procedure 338
sp_iqdroplogin system procedure 342
sp_iqemptyfile system procedure 343
sp_iqstdbspaces system procedure 344
sp_iqfile system procedure 345
sp_iqmodifylogin 349
sp_iqmodifypassword system procedure 349
sp_iqobjectinfo system procedure 349
sp_iqpassword system procedure 361
sp_iqspaceused system procedure 352
sp_iqsysmon system procedure 354
sp_objectpermission system procedure 362
sp_sys_priv_role_info 84, 365
SQL functions
AES_DECRYPT function 198
AES_ENCRYPT function 195
standalone role 3
stored procedures
granting privileges to execute 139
sp_iqbackupdetails 324
sp_iqbackupsummary 326
string comparisons
on encrypted text 218
string functions
REPLACE 196
<table>
<thead>
<tr>
<th>STRING_RTRUNCATION</th>
<th>ALTER ANY TEXT CONFIGURATION 70</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciphertext protection 219</td>
<td>ALTER ANY TRIGGER 72</td>
</tr>
<tr>
<td>database option 219</td>
<td>ALTER ANY VIEW 77</td>
</tr>
<tr>
<td>strings</td>
<td>ALTER DATABASE 42</td>
</tr>
<tr>
<td>length for database options 292</td>
<td>ALTER DATATYPE 45</td>
</tr>
<tr>
<td>replacing substrings 196</td>
<td>BACKUP DATABASE 42</td>
</tr>
<tr>
<td>STRIP</td>
<td>by functional area 41</td>
</tr>
<tr>
<td>LOAD TABLE keyword 211</td>
<td>CHANGE PASSWORD 73</td>
</tr>
<tr>
<td>STRIP option 209, 211</td>
<td>CHECKPOINT 42</td>
</tr>
<tr>
<td>subqueries</td>
<td>COMMENT ANY OBJECT 55</td>
</tr>
<tr>
<td>scalar value 137</td>
<td>CREATE ANY INDEX 50</td>
</tr>
<tr>
<td>summary 324</td>
<td>CREATE ANY MATERIALIZED VIEW 52</td>
</tr>
<tr>
<td>Sybase IQ User Administration</td>
<td>CREATE ANY OBJECT 55</td>
</tr>
<tr>
<td>sp_iqdroplogin 342</td>
<td>CREATE ANY PROCEDURE 60</td>
</tr>
<tr>
<td>SYS_AUTH_DBA_ROLE</td>
<td>CREATE ANY SEQUENCE 64</td>
</tr>
<tr>
<td>grant 32</td>
<td>CREATE ANY TABLE 67</td>
</tr>
<tr>
<td>roles granted 33</td>
<td>CREATE ANY TEXT CONFIGURATION 71</td>
</tr>
<tr>
<td>system privileges granted 34</td>
<td>CREATE ANY TRIGGER 72</td>
</tr>
<tr>
<td>SYS_AUTH_SA_ROLE</td>
<td>CREATE ANY VIEW 77</td>
</tr>
<tr>
<td>grant 28</td>
<td>CREATE DATATYPE 45</td>
</tr>
<tr>
<td>system privileges granted 29</td>
<td>CREATE EXTERNAL REFERENCE 47</td>
</tr>
<tr>
<td>SYS_AUTH_SSO_ROLE</td>
<td>CREATE MATERIALIZED VIEW 52</td>
</tr>
<tr>
<td>grant 31</td>
<td>CREATE MESSAGE 53</td>
</tr>
<tr>
<td>system privileges granted 32</td>
<td>CREATE PROCEDURE 60</td>
</tr>
<tr>
<td>SYS_RUN_REPLICATION_ROLE</td>
<td>CREATE PROXY TABLE 68</td>
</tr>
<tr>
<td>grant 26</td>
<td>CREATE TABLE 68</td>
</tr>
<tr>
<td>SYSCOLAUTH view</td>
<td>CREATE TEXT CONFIGURATION 71</td>
</tr>
<tr>
<td>privileges 83</td>
<td>CREATE VIEW 77</td>
</tr>
<tr>
<td>SYSGROUPS view</td>
<td>data types 45</td>
</tr>
<tr>
<td>privileges 83</td>
<td>database 41</td>
</tr>
<tr>
<td>SYSPROCAUTH view</td>
<td>database options 43</td>
</tr>
<tr>
<td>privileges 83</td>
<td>debugging 46</td>
</tr>
<tr>
<td>SYSTABAUTH view</td>
<td>DEBUGGING 46</td>
</tr>
<tr>
<td>privileges 83</td>
<td>DELETE ANY TABLE 68</td>
</tr>
<tr>
<td>system privilege</td>
<td>DROP ANY INDEX 51</td>
</tr>
<tr>
<td>grant 81</td>
<td>DROP ANY MATERIALIZED VIEW 53</td>
</tr>
<tr>
<td>granting 274</td>
<td>DROP ANY OBJECT 56</td>
</tr>
<tr>
<td>revoke 82</td>
<td>DROP ANY PROCEDURE 61</td>
</tr>
<tr>
<td>revoking 286</td>
<td>DROP ANY SEQUENCE 64</td>
</tr>
<tr>
<td>system privileges 41</td>
<td>DROP ANY TABLE 69</td>
</tr>
<tr>
<td>ACCESS SERVER LS 58</td>
<td>DROP ANY TEXT CONFIGURATION 72</td>
</tr>
<tr>
<td>alphabetical listing 78</td>
<td>DROP ANY VIEW 78</td>
</tr>
<tr>
<td>ALTER ANY INDEX 50</td>
<td>DROP CONNECTION 43</td>
</tr>
<tr>
<td>ALTER ANY MATERIALIZED VIEW 52</td>
<td>DROP DATATYPE 45</td>
</tr>
<tr>
<td>ALTER ANY OBJECT 54</td>
<td>DROP MESSAGE 53</td>
</tr>
<tr>
<td>ALTER ANY OBJECT OWNER 55</td>
<td>events 47</td>
</tr>
<tr>
<td>ALTER ANY PROCEDURE 59</td>
<td>EXECUTE ANY PROCEDURE 61</td>
</tr>
<tr>
<td>ALTER ANY SEQUENCE 64</td>
<td></td>
</tr>
</tbody>
</table>
Index

external environment 47
files 48
indexes 50
insert any table 69
ldap 51
list 275, 287
load any table 69
manage any dbspace 46
manage any event 47
manage any external
  environment 48
manage any external object 48
manage any ldap server 51
manage any login policy 73
manage any mirror server 58
manage any object privileges 57
manage any spatial objects 66
manage any statistics 66
manage any user 73
manage any web service 78
manage auditing 61
manage multiplex 59
manage profiling 43
manage replication 62
manage roles 63
materialized views 51
messages 53
mirror server 58
miscellaneous 54
monitor 43
multiplex 58
procedures 59
read client file 49
read file 49
reorganize any object 57
replication 61
roles 62
select any table 69
sequences 64
server 65
server operator 65
set any public option 44
set any security option 44
set any system option 44
set any user defined option 44
set user 74
spatial objects 66
statistics 66
tables 67
text configurations 70
triggers 72
truncate any table 70
update any table 70
upgrade role 63
use any sequence 65
valid any object 57
views 76
web services 78
write client file 49
write file 49
system procedures
  sp_expireallpasswords 320
  sp_iqaddlogin 323
  sp_iqbackupdetails 324
  sp_iqbackupsummary 326
  sp_iqconnection 327
  sp_iqcopyloginpolicy 331, 348
  sp_iqdbspaceobjectinfo 338
  sp_iqdroplogin 342
  sp_iqemptyfile 343
  sp_iqestdbspaces 344
  sp_iqfile 345
  sp_iqmodifylogin 349
  sp_iqobjectinfo 349
  sp_iqpassword 361
  sp_iqspaceused 352
  sp_iqsysmon 354
system role
dbo 22
diagnostics 23
public 23
revoke 27
sys 24
sys_repllication_admin_role 25
sys_spatial_admin_role 27
system roles 22
  rs_systabgroup 24
system secure feature 160
system tables
  privileges 83
  users and groups 83
system views
  privileges 83
sysuserauth view
  privileges 83
sysusertlist view
  privileges 83
SYSUSERPERMS view
privileges 83

T
table
  grant LOAD privilege 87
  grant TRUNCATE privilege 89
table and views
  grant ALTER privilege 86
  grant DELETE privilege 86
  grant INSERT privilege 87
  grant REFERENCES privilege 88
  grant SELECT privilege 88
  grant UPDATE privilege 90	
tables
  loading 200
  moving to new dbspace 94
  owner 84
  qualified names 37
  role owners 37
task-based security restrictions 139
TDS communication parameter 374
trailing blanks
  trimming 209, 211
transaction management
  monitoring with sp_iqsysmon 354
trimming trailing blanks 209, 211
TRUNCATE privilege, table
  grant 89
TRUSTED_CERTIFICATES_FILE
  disable 166
  enable 166
TRUSTED_CERTIFICATES_FILE option 298

U
UPDATE privilege, tables and views
  grant 90
USAGE privilege, sequence generator
  grant 92
USAGE statement
  grant 277
  revoke 289
user accounts
  unlock 127
user administration
  See login management
user defined role
  converting 5
  extending 5
user IDs
  case sensitivity 122
  changing passwords 262
  creating 123
  listing 83
user-defined role
  add 7
  create 4
  delete 9
  drop 9
  remove membership 9
user-user 121
users 119
  adding 323
  altering 242
  creating 255
  delete 123
  dropping 260, 279, 342
  locking 131
  locking out 125
  login failures 132
  modifying 349
  unlocking 126
USING
  LOAD TABLE keyword 208
USING FILE clause
  LOAD TABLE statement 208
util_db.ini file 158
utility database
  connecting 159
  password to create databases 159
  security 158
  setting password 158
  starting 158
V
VALIDATE LDAP SERVER statement 294
VERIFY_PASSWORD_FUNCTION option 299
verifying passwords 124
views
  deleting 138
  inserting and deleting 138
  owner 84
  security 136
  SELECT statement restrictions 137
  using 137
Index

W
WITH GRANT OPTION clause 90

WORD SKIP option
LOAD TABLE statement 214