SECURITY GUIDE | PUBLIC
SAP Adaptive Server Enterprise 16.0 SP03
Document Version: 1.0 – 2020-03-04

Security Administration Guide

THE BEST RUN SAP

Content

1 Introduction to Security

Information is possibly your company's greatest asset. Information needs protection just like any other asset. As a system administrator, determine how best to protect the information contained in company databases, and who may access the information. Individual database servers need strong, yet flexible, security support.

1.1 What is “Information Security?”

Users and the data they access may be located anywhere in the world, connected by untrusted networks. Ensuring the confidentiality and integrity of sensitive data and transactions in this environment is critical.

Information is useful only if it gets to the people who need it, when they need it. With complex and dynamically changing business relationships, it is critical that information gets only to authorized users.

These are some general guidelines when considering security for your enterprise:

- Sensitive information should be kept confidential – determine which users should have access to what information.
- The system should enforce integrity – the server should enforce rules and constraints to ensure that information remains accurate and complete.
- The information should be available – even with all the safeguards in place, anybody who needs access to the information should have it available when the information is needed.

Identify what is it that your organization wants to protect, and what the outside world requires from your organization:

- Identify the information assets and the security risks associated with them if they become vulnerable or compromised.
- Identify and understand any laws, statutes, regulations, and contractual agreements that apply to your organization and the information assets.
- Identify your organization’s business processes and the requirements they impose on information assets, to balance practical considerations with the security risks.

Security requirements change over time. Periodically reassess security requirements to make sure they still reflect your organization’s needs.

Set up a series of controls and policies that meet the company's security objectives, the result of which is an information security policy document that clarifies decisions made for information security.

1.2 Common Criteria Configuration Evaluation

SAP Adaptive Server Enterprise (SAP ASE) has been evaluated and validated in accordance with the provisions of the Common Criteria Evaluation and Validation Scheme. SAP ASE also uses FIPS 140-2 certified modules for implementing encryption functionality.

Common Criteria for Information Technology Security Evaluation is an international standard (ISO/IEC 15408) for computer security certification. Common Criteria is developed by the governments of Canada, France, Germany, Netherlands, UK and the United States.

SAP ASE version 15.0.1 completed Common Criteria validation in September, 2007. The Evaluated configuration consists of version 15.0.1 with the security and directory services option. The evaluation for security was carried out in accordance with the Common Criteria Evaluation and Validation Scheme (CCEVS) process and scheme. The criteria against which SAP ASE was judged are described in the Common Criteria for Information Technology Security Evaluation, Version 2.3 and International Interpretations effective on August, 2005. If you configure SAP ASE as specified in the *Supplement for Installing Adaptive Server for Common Criteria Configuration*, SAP ASE satisfies all of the security functional requirements stated in the Adaptive Server Enterprise Security Target (Version 1.5).

SAP ASE supports eight security functions:

- Cryptographic support – transparent encryption of data is supported at the column level. SQL statements and extensions provide secure key management.
- Security audit – an audit mechanism that checks access, authentication attempts, and administrator functions. The security audit records the date, time, responsible individual, and other details describing the event in the audit trail.
- User data protection – discretionary access control policy over applicable database objects is provided for these objects: databases, tables, views, stored procedures, and encryption keys.
- Identification and authentication – an identification and authentication mechanism is provided in addition to the underlying operating system mechanism.
- Security management – functions that allow you to manage users and associated privileges, access permissions, and other security functions such as the audit trail are provided. These functions are restricted based on discretionary access control policy rules, including role restrictions.
- Protection of the TOE Security Function (TSF) – SAP ASE keeps its context separate from that of its users, and uses operating system mechanisms to ensure that memory and files used by SAP ASE have the appropriate access settings. SAP ASE interacts with users through well-defined interfaces designed to ensure that its security policies are enforced.
- Resource utilization – resource limits are provided to prevent queries and transactions from monopolizing server resources.
- Target of Evaluation (TOE) access – authorized administrators are allowed to construct login triggers that restrict logins to a specific number of sessions and restrict access based on time. Authorized administrators can also restrict access based on user identities.

1.3 FIPS 140-2 Validated Cryptographic Module

SSL is the standard for securing the transmission of sensitive information, such as credit card numbers, stock trades, and banking transactions over the Internet.

SAP ASE uses the proprietary SAP Common Crypto Library, as the SSL provider and cryptography library. Beginning with SAP ASE version 16.0 SP03, OpenSSL is not supported for encryption and decryption.

SAP Common Crypto Library is FIPS 140-2 compliant. For information about FIPS certification, see the *Computer Security Resource Center* at https://csrc.nist.gov.

The encrypted columns feature relies on symmetric-key cryptography, and uses the same FIPS 140-2 validated cryptographic modules as SSL. See *Database Encryption*.

A Security and Directory Services license is required to use SSL and to enable the `FIPS login password encryption` parameter. You must have an encrypted columns license to use the encrypted columns feature.

2 Security Administration in SAP ASE

Perform major tasks to securely administer SAP ASE.

1. Install SAP ASE, including auditing – includes preparing for installation, loading files from your distribution medium, performing the actual installation, and administering required physical resources. See the installation documentation for your platform and *Auditing*.
2. Set up a secure administrative environment – Set up system administrators and system security officers, create login profiles and establish password and login policies. See *Managing Logins and Database Users*.
3. Set up logins, database users and roles – Add user logins to the server and assign login profiles to them. Create user defined roles, define role hierarchies and mutual exclusivity of roles, and assign roles to logins. Add users to databases. See *Managing Logins and Database Users* and *Create a User-Defined Role*.
4. Administer permissions for users, groups, and roles – Grant and revoke permissions for certain SQL commands, executing certain system procedures, and accessing databases, tables, particular table columns, and views. Create access rules to enforce fine-grained access control. See *Managing User Permissions*.
5. Configure encryption in your database to encrypt sensitive data in tables. Encrypt sensitive data – Configure SAP ASE to use column-level encryption, decide which columnar data to encrypt, perform a one-time key creation operation, and use `alter table` to perform initial data encryption. See *Database Encryption*.
6. Establish integrity controls over data – Add check constraints, domain roles, and referential constraints to validate incoming data. See *Transact-SQL Users Guide* and *Reference Manual: Commands*
7. Set up and maintain auditing – Determine what is to be audited, audit the use of SAP ASE, and use the audit trail to detect penetration of the system and misuse of resources. See *Auditing*.
8. Set up your installation for advanced authentication mechanisms and network security – Configure the server to use services, such as LDAP, PAM, or Kerberos- based user authentication, data confidentiality with encryption, data integrity. See *External Authentication* and *Confidentiality of Data*.

Related Information

Auditing [page 334]
Manage SAP ASE Logins and Database Users [page 25]
Create a User-Defined Role [page 98]
Manage User Permissions [page 167]
External Authentication [page 116]
Confidentiality of Data [page 289]

2.1 Recommendations for Setting up Security

Security recommendations for setting up logins and auditing.

- Using the sa login – when you install SAP ASE, a single login called "sa" is configured with the system administrator and system security officer roles, which means that the sa login has unlimited control over what occurs in the database.
  Use the sa login only during initial setup. Instead of allowing several users to use the sa account, establish individual accountability by assigning specific roles to individual administrators.
- Changing the sa login password – the sa login is configured initially with a NULL password. Use `alter login` to change the password immediately after installation.

  > ⚠ Caution
  >
  > When logging in to SAP ASE, do not use the `-P` option of `isql` to specify your password because another user may have an opportunity to see it.

- Enabling auditing – enable auditing early in the administration process so that you have a record of privileged commands that are executed by system security officers and system administrators. You might also want to audit commands that are executed by those with other special roles, such as operators when they dump and load databases
- Assigning login names – assign login names that are the same as their respective operating system login names. This makes logging in easier, simplifies management of server and operating system login accounts, and makes it easier to correlate the audit data generated with that of the operating system.

2.1.1 Example of Setting up Security

Security roles and specific commands are used to set up a secure operating environment.

This table shows special roles assigned to users.

Table 1: Users to whom you assign roles

Name	Privilege	Operating system login name
Rajnish Smith	sso_role	rsmith
Catherine Macar-Swan	sa_role	cmacar
Soshi Ikedo	sa_role	sikedo
Julio Rozanski	oper_role	jrozan
Alan Johnson	dbo	ajohnson

This table shows the sequence of commands you might use to set up a secure operating environment for SAP ASE, based on the role assignments listed above. After logging in to the operating system, issue these commands using the initial sa account. .

Table 2: Examples of commands used to set up security

Commands	Result
`isql -Usa`	Logs in to SAP ASE as "sa." Both sa_role and sso_role are active.

`sp_audit "security", "all", "all", "on"`
`sp_audit "all", "sa_role", "all", "on"`
`sp_audit "all", "sso_role", "all", "on"`
	Sets auditing options for server-wide, security-relevant events, and the auditing of all actions that have sa_role or sso_role active.

`sp_configure "auditing", 1`	Enables auditing.
i Note Before you enable auditing, set up a threshold procedure for the audit trail and determine how to handle the transaction log in `sybsecurity`.	
`create login`	Adds logins and passwords.
`grant role`	Grant roles.
`use sybsecurity` `sp_changedbowner rsmith`	Grants access to the auditing database, `sybsecurity`, by making Rajnish, who is the system security officer, the database owner. Alan is not granted any system-defined roles.
`sp_locklogin sa, "lock"`	Locks the "sa" login so that no one can log in as "sa." Individuals can assume only the roles that are configured for them.
i Note Do not lock the "sa" login until you have granted individual users the sa_role and sso_role roles and have verified that the roles operate successfully.	

2.2 Security Features in SAP ASE

SAP ASE provides security features that aid in protecting sensitive data.

- *Identification and Authentication* – ensures that only authorized users can log in to the system. In addition to password-based login authentication, SAP ASE supports external authentication using Kerberos, LDAP, or PAM.
- *Discretionary Access Control* – provides access controls that give object owners the ability to restrict access to objects, usually with the `grant` and `revoke` commands. This type of control is dependent on an object owner's discretion.

- *Division of Roles* – allows an administrator to grant privileged roles to specified users so only designated users can perform certain tasks. SAP ASE has predefined roles, called "system roles," such as system administrator and system security officer. In addition, system security officers are allowed to define additional roles, called "user-defined roles."
- *Auditing for Accountability* – provides the ability to audit events such as logins, logouts, server start operations, remote procedure calls, accesses to database objects, and all actions performed by a specific user or with a particular role active. SAP ASE also provides a single option to audit a set of server-wide security-relevant events.
- *Confidentiality of Data* – maintains confidentiality of data using encryption for client/server communication, available with Kerberos or SSL. Column-level encryption preserves confidentiality of data stored in the database. Inactive data is kept confidential with a password-protected database backup.

2.2.1 Identification and Authentication

Both internal and external mechanisms can be used for authentication.

SAP ASE uses the server user identity (SUID) to uniquely identify a user with a login account name. This identity is linked to a particular user identity (UID) in each database. Access controls use the server user identity when determining whether to allow access for the user with this SUID to an object. Authentication verifies that a user is actually the person he or she claims to be.

External Authentication

Security is often enhanced in large, heterogeneous applications by authenticating logins with a central repository. SAP ASE supports a variety of external authentication methods:

- Kerberos – provides a centralized and secure authentication mechanism in enterprise environments that includes the Kerberos infrastructure. Authentication occurs with a trusted, third-party server called a key distribution center to verify both the client and the server.
- LDAP user authentication – Lightweight Directory Access Protocol (LDAP) provides a centralized authentication mechanism based on the user login name and password.
- PAM user authentication – Pluggable Authentication Module (PAM) provides a centralized authentication mechanism that uses operating system interfaces for both administration and runtime application operations.

Managing Remote Servers

Internal mechanisms for administering logins and users between servers are described in *Managing Remote Servers* in the *System Administration Guide, Volume I*

2.2.2 Discretionary Access Control

With discretionary access control, you can give various permissions to users, groups, and roles.

Object owners can grant access to the objects they own to other users. Object owners can also grant other users the ability to pass the access permission to other users. Use the `grant` command to give permissions to users, groups, and roles. Use the `revoke` command to rescind these permissions. The `grant` and `revoke` commands give users permission to execute specified commands, and to access specified tables, procedures, views, encryption keys, and columns.

Some commands can be used at any time by any user, with no permission required. Others can be used only by users of a certain status, such as a system administrator, and are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's status (as system administrator, system security officer, database owner, or database object owner), and whether a particular user is granted a permission with the option to grant that permission to other users.

Row-level Access Control

Administrators define access rules that are based on the value of individual data elements, and the server transparently enforces these rules. Once an administrator defines an access rule, it is automatically invoked whenever the affected data is queried through applications, ad hoc queries, stored procedures, views, and so on.

Using a rule-based access control simplifies both the security administration of an installation and the application development process because the server, rather than the application, enforces security. These features allow you to implement row-level access control:

- Access rules
- Application context facility
- Login triggers
- Domain integrity rules

Predicated Privileges

With predicated privileges, the table owner provides row-level access to users, groups or roles by specifying a SQL `where` on the `grant` statement. You use the full power of SQL, including access to other tables, to implement a comprehensive row-level security policy.

2.2.3 Division of Roles

The roles supported by SAP ASE enable you to enforce and maintain individual accountability. SAP ASE provides system roles, such as system administrator and system security officer, and user-defined roles, which are created by a system security officer.

Roles are collections of privileges that allow the role assignee to do their job. Roles provide individual accountability for users performing operational and administrative tasks, and allow you to audit and attribute actions to these users.

Role Hierarchy

A system security officer can define role hierarchies such that if a user has one role, the user automatically has roles lower in the hierarchy. For example, the "chief_financial_officer" role might contain both the "financial_analyst" and the "salary_administrator" roles. The chief financial officer can perform all tasks and see all data that can be viewed by salary administrators and financial analysts.

Restrictions on Role Activation

The Security Administrator restricts the conditions under which a role may be activated by specifying a `where` clause on the `grant role` statement. SAP ASE evaluates the condition expressed by the `where` clause when the role is activated, either automatically during login, or when processing the `set role` statement.

Mutual Exclusivity

You can define roles to be mutually exclusive either at the membership level, or at the activation level. For example:

- You may not want to grant both the "payment_requestor" and "payment_approver" roles to the same user.
- A user might be granted both the "senior_auditor" and the "equipment_buyer" roles, but you may not want to permit the user to have both roles enabled at the same time.

You can define system roles, as well as user-defined roles, to be in a role hierarchy or to be mutually exclusive. For example, you might want a "super_user" role to contain the system administrator, operator, and technical support roles. Additionally, you may want to define the system administrator and system security officer roles to be mutually exclusive for membership; that is, a single user cannot be granted both roles.

2.2.4 Auditing for Accountability

SAP ASE includes a comprehensive auditing system.

The auditing system consists of:

- The `sybsecurity` database
- Configuration parameters for managing auditing
- `sp_audit` to set all auditing options
- `sp_addauditrecord` to add user-defined records to the audit trail

When you install auditing, you can specify the number of audit tables that SAP ASE uses for the audit trail. If you use two or more tables to store the audit trail, you can set up a smoothly running audit system with no manual intervention and no loss of records.

A system security officer manages the audit system and is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a system security officer, you can establish auditing for events such as:

- Server-wide, security-relevant events
- Creating, dropping, and modifying database objects
- All actions by a particular user or all actions by users with a particular role active
- Granting or revoking database access
- Importing or exporting data
- Logins and logouts
- All actions related to encryption keys

2.2.5 Confidentiality of Data

You can maintain the confidentiality of data by encrypting client-server communications by using the Secure Sockets Layer (SSL) standard or Kerberos.

You can protect the confidentiality of data by using column-level encryption in the database and encrypting backups for offline data. The `dump` and `load database` commands include a `<password>` parameter that allows you to password-protect your database dumps. For more information see *Database Encryption*, *Backing Up and Restoring User Databases*, in *System Administration Guide: Volume 2*.

2.3 Manage Deployed Source in Source Code Control Systems

SAP recommends that all user created source files associated with SAP ASE deployment be managed with a third party source control system (SCCS, RCS, Clearcase, Perforce, and so on).

It is further recommended that deployment of these files be done in an automated way such as use of a bill of materials and scripts to install the files in a reliable and repeatable way. The script source itself is recommended to be maintained within the source control system.

3 Data Protection in SAP ASE

SAP ASE provides the technical enablement and infrastructure to allow you run applications on SAP ASE to conform to the legal requirements of data protection in the different scenarios in which SAP ASE is used.

Introduction to Data Protection

Data protection is associated with numerous legal requirements and privacy concerns. In addition to compliance with general data privacy acts, it is necessary to consider compliance with industry-specific legislation in different countries. This section describes the specific features and functions that SAP ASE provides to support compliance with the relevant legal requirements and data privacy.

This section and any other sections in this guide do not give any advice on whether these features and functions are the best method to support company, industry, regional or country-specific requirements. Furthermore, this guide does not give any advice or recommendations with regard to additional features that would be required in a particular environment; decisions related to data protection must be made on a case-by-case basis and under consideration of the given system landscape and the applicable legal requirements.

> **i** Note
>
> In the majority of cases, compliance with data privacy laws is not a product feature. SAP software supports data privacy by providing security features and specific functions relevant to data protection, such as functions for the simplified blocking and deletion of personal data. SAP does not provide legal advice in any form. The definitions and other terms used in this guide are not taken from any given legal source.

Table 3: Glossary

Term	Definition
Blocking	A method of restricting access to data for which the primary business purpose has ended.
Business Purpose	A legal, contractual, or in other form justified reason for the processing of **personal data**. The assumption is that any purpose has an end that is usually already defined when the purpose starts.
Consent	The action of the data subject confirming that the usage of his or her personal data shall be allowed for a given purpose. A consent functionality allows the storage of a consent record in relation to a specific purpose and shows if a data subject has granted, withdrawn, or denied consent.
Deletion	Deletion of **personal data** so that the data is no longer available.

Term	Definition
End of business	Date where the business with a data subject ends, for example the order is completed, the subscription is canceled, or the last bill is settled.
End of purpose (EoP)	End of purpose and start of blocking period. The point in time, when the primary processing purpose ends (e.g. contract is fulfilled).
End of purpose (EoP) check	A method of identifying the point in time for a data set when the processing of **personal data** is no longer required for the primary **business purpose**. After the **EoP** has been reached, the data is **blocked** and can only be accessed by users with special authorization (for example, tax auditors).
Personal data	Any information relating to an identified or identifiable natural person ("data subject"). An identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural, or social identity of that natural person.
Residence period	The period of time between the end of business and the end of purpose (EoP) for a data set during which the data remains in the database and can be used in case of subsequent processes related to the original purpose. At the end of the longest configured residence period, the data is blocked or deleted. The residence period is part of the overall retention period.
Retention period	The period of time between the end of the last business activity involving a specific object (for example, a business partner) and the deletion of the corresponding data, subject to applicable laws. The retention period is a combination of the residence period and the blocking period.
Sensitive personal data	A category of personal data that usually includes the following type of information: • Special categories of personal data, such as data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, genetic data, biometric data, data concerning health or sex life or sexual orientation, or personal data concerning bank and credit accounts. • Personal data subject to professional secrecy • Personal data relating to criminal or administrative offenses • Personal data concerning insurances and bank or credit card accounts

Term	Definition
Where-used check (WUC)	A process designed to ensure data integrity in the case of potential blocking of business partner data. An application's where-used check (WUC) determines if there is any dependent data for a certain business partner in the database. If dependent data exists, this means the data is still required for business activities. Therefore, the blocking of business partners referenced in the data is prevented.

SAP ASE Approach to Data Protection

Many data protection requirements depend on how the business semantics or context of the data stored and processed in SAP ASE are understood.

> **i** Note
>
> Using capabilities to communicate with other data sources, SAP ASE may also be used to process data that is stored in other systems and accessed through virtual tables.

In SAP ASE installations, the business semantics of data are part of the application definition and implementation. SAP ASE provides the features for working with technical database objects, such as tables. It is therefore the application that "knows," for example, which tables in the database contain sensitive personal data, or how business level objects, such as sales orders, are mapped to technical objects in the database. Applications built on top of SAP ASE need to make use of features provided by SAP ASE to implement compliance requirements for their specific use case.

SAP ASE provides a variety of security-related features to implement general security requirements that are also required for data protection and privacy:

Aspect of Data Protection and Privacy	SAP ASE Feature	More Information

Access control	Several features in SAP ASE provide access control:
	• Authentication
	• Authorization
	• Data encryption:
	  ○ Encrypted Columns for encrypting table columns
	  ○ Database Encryption for encrypting all data, indexes, and transaction logs in the database at rest
	  ○ Backup (dump) and Restore (load) of databases with password protection and fully-encrypted database backups
		• Section Manage SAP ASE Logins and Database Users [page 25]
		• Section Manage Roles [page 93]
		• Section Manage User Permissions [page 167]
		• Section Confidentiality of Data [page 289]
		• Document *Database Encryption*

Aspect of Data Protection and Privacy	SAP ASE Feature	More Information
Access logging	Auditing	• Section Auditing [page 334] • Section *Logging* in the *SAP Adaptive Server Enterprise Cockpit Guide*
Transmission control/communication security	Support for encrypted communication on all internal and external channels	• Section External Authentication [page 116] • Section SSL Overview [page 291] • Section Kerberos Confidentiality [page 323] • Section On-Demand Encryption [page 320]
Availability control	Several features in SAP ASE provide availability control: • Backup and Recovery • High Availability with Failover • High Availability and Disaster Recovery (HADR)	• Section *Mirroring Database Devices* in the *SAP ASE System Administration Guide: Volume 2* • Section *Developing a Backup and Recovery Plan* in the *SAP ASE System Administration Guide: Volume 2* • Section *Backup Up and Restoring User Databases* in the *SAP ASE System Administration Guide: Volume 2* • Section *Restoring the System Databases* in the *SAP ASE System Administration Guide: Volume 2* • Section *Archive Database Access* in the *SAP ASE System Administration Guide: Volume 2* • Document *Using SAP Failover in a High Availability System* • Document *HADR Users Guide*
Separation by purpose	Separation by purpose is subject to the organizational model implemented and must be applied as part of the authorization concept. Data separation can be achieved in SAP ASE using: • Multiple databases • Separating ownership by database user	• Section Manage User Permissions [page 167] • Section Granular Permissions [page 241]

> ⚠ Caution
>
> The extent to which data protection is ensured depends on secure system operation. Network security, security note implementation, adequate logging of system changes, and appropriate usage of the system are the basic technical requirements for compliance with data privacy legislation and other legislation.

3.1 Deletion of Personal Data

SAP ASE supports the deletion of data in tables using SQL deletion commands. Applications running on SAP ASE must make use of such commands to implement deletion requirements of personal data.

End of purpose checks, including implementation of legally required retention periods and data blocking, are managed by the application. Applications can implement data blocking using SAP ASE mechanisms such as authorization and table creation. For example, an application could transfer blocked data to separate database tables that are protected by special authorizations.

Once data has been deleted, the delete operation cannot be undone using SQL statements.

> ℹ Note
>
> Following standard practice, deletion of personal data is not enforced in backups.

4 Manage SAP ASE Logins and Database Users

The responsibility of adding new login accounts, adding users to databases, and granting users permission to use commands and access database objects is divided among the system security officer, system administrator, and database owner.

> **i** Note
>
> Permission requirements for operations assume that granular permissions is disabled. Operations may differ when granular permissions is enabled.

Table 4: Managing Users in SAP ASE

Task	Required role	Command or procedure	Database, group, or role
Create login accounts	System security officer	`create login`	Master database
Alter login accounts	System security officer The exception is that users can change their own password and full name.	`alter login`	Master database
Drop login accounts	System security officer	`drop login`	Master database
Create groups	Database owner or system administrator	`sp_addgroup`	User database
Create and assign roles	System security officer	`create role`, `grant role`	Master database
Add users to database and assign groups	Database owner or system administrator	`sp_adduser`	User database
Alias users to other database users	Database owner or system administrator	`sp_addalias`	User database
Grant groups, users, or roles permission to create or access database objects and run commands	Database owner, system administrator, system security officer, or object owner	`grant`	User database

Related Information

Granular Permissions [page 241]

4.1 Introduction to Logins

A login defines a name and a password for a user to allow access to SAP ASE.

When you execute `create login`, a row is added to `master.dbo.syslogins`, a unique system user ID (`suid`) is assigned for the new user, and specified attribute information is filled in. When a user logs in, SAP ASE looks in `syslogins` for the name and password provided by the user. The `password` column is encrypted with a one-way algorithm so it is not readable.

A login profile is a collection of attributes to be applied to a set of login accounts. The attributes define login characteristics, such as default roles or the login script associated with each login bound to the profile. Login profiles save time for the system security administrator because attributes of login accounts are set up and managed in one place.

4.1.1 Create Login Accounts

Use `create login` to add a new login name. Only the system security officer can execute `create login`.

The following steps describe creating a login account for a particular server and manage permissions for the users:

1. A system security officer creates a login account for a new user.
2. A system administrator or database owner adds a user to database or assigns a user to a group.
3. A system security officer grants specific roles to the user.
4. A system administrator, database owner, or object owner grants the user, or group specific permissions on specific commands and database objects.

See `create login` in the *Reference Manual: Commands* for complete syntax.

At login creation, the `crdate` column in `syslogins` is set to the current time.

The `suid` column in `syslogins` uniquely identifies each user on the server. A user's `suid` remains the same, no matter what database he or she is using. The `suid` 1 is always assigned to the default "sa" account that is created when SAP ASE is installed. Other users' server user IDs are integers assigned consecutively each time `create login` is executed.

The following statement sets up an account for the user "maryd" with the password "100cents," the default database (`master`), the default language (us_english), and no full name:

```
create login maryd with password "100cents"
```

The password requires quotation marks because it begins with 1.

After this statement is executed, "maryd" can log in to the server. She is automatically treated as a "guest" user in `master`, with limited permissions, unless she has been specifically given access to `master`.

The following statement sets up a login account "omar_khayyam" and password "rubaiyat" and makes "pubs2" the default database for this user:

```
create login omar_khayyam with password rubaiyat default database pubs2
```

4.1.2 Last Login and Managing Inactive Accounts

Manage and track SAP ASE accounts.

SAP ASE provides security for user accounts by:

- Tracking the creation date.
- Recording the last login time for an account.
- Determining which accounts are stale and locked due to inactivity.
- Recording the reason an account is locked, when the account is locked, and the identity of the user who locked the account.

4.1.2.1 Defining a Stale Period

`stale period` is an attribute of login profiles which indicates the duration a login account is allowed to remain inactive before being locked due to inactivity.

If the login profile `track lastlogin` attribute is not set to `0` and the login account is not exempt from locking due to inactivity, then the `syslogins.lastlogindate` and `syslogins.pwdate` fields are checked to determine inactivity during the login process or during the execution of `sp_locklogin`.When login accounts are locked due to inactivity, the `locksuid`, `lockreason` and `lockdate` fields in `syslogins` will be set as follows:

Value of lockreason	Value for locksuid	Description of lockreason of account
4	NULL	Account locked automatically due to inactivity.

If a High Availability solution is setup, the `syslogins.lastlogindate` and `syslogins.pwdate` are synchronized on both the servers. Login accounts locked on one server are also locked on the companion server.

4.1.2.2 Tracking the Last Login

Tracking of the last login date time can be set through the `track lastlogin` attribute of a login profile.

```
create login profile general_lp with track lastlogin true authenticate with ASE
```

4.1.2.3 Preventing Inactive Accounts from Being Locked

Login accounts can be set to be exempt login from being locked due to inactivity be using the `exempt inactive lock` clause.

The following statement creates the login account "user33" which is exempt from being lock due to inactivity.

```
create login user33 with password AT0u7gh9wd exempt inactive lock true
```

4.1.3 Authentication Mechanisms for Login

The supported authentication mechanisms are: `ASE`, `LDAP`, `PAM`, `KERBEROS`, and `ANY`.

When `ANY` is used, SAP ASE checks for a defined external authentication mechanism. If one is defined, SAP ASE uses the defined mechanism., otherwise the `ASE` mechanism is used.

4.1.4 Change Login Accounts

Use `alter login` to add, drop or change attributes of a login and their corresponding values.

`alter login` allows you to:

- Add or drop auto activated roles
- Change a password
- Change the login profile association
- Change or add a full name
- Specify the password expiration and the minimum password length
- Specify the maximum failed attempts
- Specify an authentication mechanism
- Specify the default language and default database
- Invoke a login script
- Exempt inactive login accounts

A system administrator can use `alter login` to set password length and expiration, to limit failed login attempts, drop attributes, and to specify that a login script be run automatically when a user logs in.

After you execute `alter login` to change the default database, the user is connected to the new default database the next time he or she logs in. However, `alter login` does not automatically give the user access to the database. Unless the database owner has been assigned access with `sp_adduser`, `sp_addalias`, or with a guest user mechanism, the user is connected to `master` even after his or her default database has been changed.

This example changes the default database for the login account anna to `pubs2`:

```
alter login anna modify default database pubs2
```

This example changes the default language for claire to French:

```
alter login claire modify default language french
```

4.1.5 Drop Login Accounts

The command `drop login` removes a user login by deleting the user's entry from `master.dbo.syslogins`.

You cannot drop a login who is a user in any database, and you cannot drop a user from a database if the user owns any objects in that database or has granted any permissions on objects to other users. The dropped login account's server user ID (suid) can be reused when the next login account is created. This only occurs when the dropped login holds the highest suid in `syslogins` , but could compromise accountability if execution of `drop login` is not being audited. You cannot drop the last remaining System Security Officer's or System Administrator's login account.

The `with override` clause drops the login even if there are non-available databases that cannot be checked for login references.

The following example drops login accounts `mikeb` and `rchin`.

```
drop login mikeb, rchin
```

See the *Reference Manual: Commands* for complete `drop login` syntax.

4.2 Choose and Create a Password

The system security officer assigns each user a password using `create login` when adding the users. Users can modify their passwords at any time using the `alter login` statement.

When you create your password:

- Do not use information such as your birthday, street address, or any other word or number that has anything to do with your personal life.
- Do not use names of pets or loved ones.
- Do not use words that appear in the dictionary or words spelled backwards.

The most difficult passwords to guess are those that combine uppercase and lowercase letters and numbers. Never give anyone your password, and never write it down where anyone can see it.

Passwords must:

- Be at least 6 characters long. SAP recommends passwords of 8 characters or longer.
- Consist of any printable letters, numbers, or symbols. The complete special character set is: !@#$ %^&*()_-+={[}]|\:;"'<,>.?/~`
- Be enclosed in quotation marks in `create login` if they:
  - Include any character other than A – Z, a – z, 0 – 9,_, #, valid single-byte or multibyte alphabetic characters, or accented alphabetic characters

- Begin with a number 0 – 9

Related Information

Password Complexity Checks [page 36]

4.2.1 Maximum Login Attempts

Setting the maximum number of login attempts allowed provides protection against "brute-force" or dictionary-based attempts to guess passwords.

A system security officer can specify a maximum number of consecutive login attempts allowed, after which the login or role is automatically locked. The number of allowable failed login attempts can be set for the entire server, or for individual logins and roles. Individual settings override the server-wide setting.

The number of failed logins is stored in the `logincount` column in `master..syslogins`. A successful login resets the number of failed logins to 0.

Setting the Server-Wide Maximum Failed Logins

Use `sp_passwordpolicy` to set server-wide maximum number of failed logins for logins and roles.

To set the number of failed logins allowed, enter:

```
sp_passwordpolicy 'set', 'maximum failed logins', <'number'>
```

By default, `maximum failed logins` is turned off and this check is not applied to passwords.

See `sp_passwordpolicy` in *Reference Manual: Procedures*.

Setting the Maximum Failed Logins for Logins

Use `create login` to set the maximum number of failed login attempts for a specific login.

This example creates the new login "joe" with the password "Djdiek3" and sets the maximum number of failed login attempts to 3:

```
create login joe with password Djdiek3 max failed attempts 3
```

See `create login` in *Reference Manual: Commands*.

Setting the Maximum Failed Logins for Roles

Use `create role` to set the maximum failed logins for a specific role.

This example creates “intern_role” with the password “temp244”, and sets the maximum failed logins for “inter_role” to 20:

```
create role intern_role with passwd "temp244", max failed_logins 20
```

See `create role` in *Reference Manual: Commands*.

Changing the Maximum Failed Logins for Logins

Use `alter login` to set or change the maximum failed logins for an existing login.

This example changes the maximum failed logins for the login “joe” to 40:

```
alter login joe modify max failed attempts 40
```

See `alter login` in *Reference Manual: Commands*.

Changing the Maximum Failed Logins for Roles

Use `alter role` to set or change the maximum failed logins for an existing role.

This example changes the maximum failed logins allowed for “physician_role” to 5:

```
alter role physician_role set max failed_logins 5
```

This example removes the overrides for the maximum failed logins for all roles:

```
alter role "all overrides" set max failed_logins -1
```

See `alter role` in *Reference Manual: Commands*.

4.2.2 Logging in After Losing a Password

Use `dataserver -p` to reset the password for sa_role and sso_role.

Context

Use the dataserver `-p login_name` parameter to specify the name of the system security officer or system administrator at the server start-up. This allows you to set a new password for these accounts if you cannot

recover a lost password. When you start the server with the `-p` parameter, SAP ASE generates, displays, and encrypts a random password and saves it in `master..syslogins` as that account's new password.

If sa_role does not have a password, and the server is started with -p sa_role, an error message is printed in the error log.

SAP recommends that you change the password for the login or role when the server is restarted.

Procedure

- Use `dataserver -p` when you have lost the password for either of these roles, that require a password to become active.

  For example, if the server is started with:

  ```
  xml:space="preserve">dataserver -p sa_role
  ```

  This message is displayed:

  ```
  xml:space="preserve">New password for role 'sa_role' : qjcdyrbfkxgyc0
  ```

4.2.3 Displaying Password Information for Logins

Display password information for logins using `sp_displaylogin`.

Procedure

- To display the login and password settings for a login, enter:

  ```
  sp_displaylogin [<loginame >[, expand_up | expand_down]]
  ```

  This example displays information about the login `joe` which is bound to a login profile:

  ```
  sp_displaylogin joe
  Suid: 3
  Loginame: joe
  Fullname: Joe Williams
  Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
  Locked: NO
  Date of Last Password Change: Sep 22 2008  3:50PM
  Password expiration interval: 0
  Password expired: NO
  Minimum password length: 6
  Maximum failed logins: 1
  Current failed login attempts: 2
  Authenticate with: ANY
  ```

```
Login Profile: emp_lp
```

This example displays information about the login `joe` which is not bound to a login profile:

```
sp_displaylogin joe
Suid: 3
Loginame: joe
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Sep 22 2008  3:50PM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 1
Current failed login attempts: 2
Authenticate with: ANY
Login Password Encryption: SHA-256
Last login date: Sep 18 2008 10:48PM
```

See `sp_displaylogin` in *Reference Manual: Procedures*.

4.2.4 Displaying Password Information for Roles

Display password information for logins using `sp_displayroles`

Procedure

- To display the login and password settings for a role, enter:

```
sp_displayroles
      [<grantee_name >[, <mode>]]
```

This example displays information about the `physician_role` role:

```
sp_displayroles physician_role, "display_info"
Role name = physician_role
Locked : NO
Date of Last Password Change : Nov 24 1997  3:35PM
Password expiration interval = 5
Password expired : NO
Minimum password length = 4
Maximum failed logins = 10
Current failed logins = 3
```

See `sp_displayroles` in *Reference Manual: Procedures*.

4.2.5 Checking Passwords for at Least One Digit

The system security officer can instruct the server to check for at least one digit in a password using the server-wide configuration parameter, `check password for digit`.

Context

If set, this parameter does not affect existing passwords. By default, checking for digits is off.

Procedure

- To active a password check for one digit, enter:

  ```
  sp_configure "check password for digit", 1
  ```

- To deactivate the password check, enter:

  ```
  sp_configure "check password for digit", 0
  ```

  See `sp_configure` in *Reference Manual: Procedures*.

4.2.6 Set or Change the Minimum Password Length

The configurable password allows you to customize passwords to fit your needs such as using four-digit personal identification numbers (PINs) or anonymous logins with NULL passwords.

> **i** Note
>
> SAP ASE uses a default value of 6 for minimum password length and recommends that you use a value of 6 or more for this parameter.

The system security officer can specify:

- A globally enforced minimum password length
- A per-login or per-role minimum password length

The per-login or per-role value overrides the server-wide value. Setting the minimum password length affects only new passwords created after setting the value.

Setting the Minimum Password Length for a Specific Login

Use `create login` to set the minimum password length for a specific login.

This creates the login "joe", with the password "Djdiek3", and sets the minimum password length to 8:

```
create login joe Djdiek3 with password @minpwdlen min password length 8
```

See `create login` in *Reference Manual: Commands*.

Setting the Minimum Password Length for a Specific Role

Use `create role` to set the minimum password length for a specific role.

This creates the role "intern_role", with the password "temp244", and sets minimum password length to 0:

```
create role intern_role with passwd "temp244", min passwd length 0
```

The original password is seven characters, but the password can be changed to any length because `min passwd length` is set to 0.

See `create role` in *Reference Manual: Commands*.

Changing the Minimum Password Length for a Specific Login

Use `alter login` to set or change minimum password length for an existing login.

This changes the minimum password length for the login "joe" to 8 characters:

```
alter login joe modify min password length 8
```

See `alter login` in *Reference Manual: Commands*.

Changing the Minimum Password Length for a Specific Role

Use `alter role` to set or change the minimum password length for an existing role.

This sets the minimum length for the role "physician_role" to 5 characters:

```
alter role physician_role set min passwd length 5
```

This overrides the minimum password length for all roles:

```
alter role "all overrides" set min passwd length -1
```

See `alter role` in *Reference Manual: Commands*.

Removing the Minimum Password Length for a Specific Login

Use `alter login` to remove the minimum password length for an existing login.

This removes any restriction to minimum password length for login "joe":

```
alter login joe modify drop min password length
```

See `alter login` in *Reference Manual: Commands*.

4.2.7 Password Complexity Checks

Use `sp_passwordpolicy` to activate password complexity checks.

Activating or Deactivating Password Complexity Checks

Login password complexity checks are extended to role passwords.

To turn off an individual password policy option, enter:

```
sp_passwordpolicy 'clear', <option>
```

To turn off all password policy options, enter:

```
sp_passwordpolicy 'clear'
```

Disallow Simple Passwords

- 0 – (default) turns off the option, and allows simple passwords.
- 1 – turns the option on, and disallows simple passwords.

For example:

```
sp_passwordpolicy 'set', 'disallow simple passwords', '1'
```

Custom Password-Complexity Checks

You can custom-configure password checking rules using `sp_extrapwdchecks` and `sp_cleanpwdchecks`.

These stored procedures are defined and located in the `master` database and are automatically invoked during password complexity checks, and when dropping a login, respectively.

4.2.7.1 Specify Characters in a Password

Use `sp_passwordpolicy` parameters to specify the minimum number of characters (digits, upper and lower characters, and so on) in a password.

- `min digits in password` – the minimum number of digits in a password. Disabled by default. Valid values are:
  - 0 through 16 – the minimum number of digits that must exist in a password.
  - -1 – the password cannot contain digits.
- `min alpha in password` – the minimum number of alphabetic characters allowed in a password. This value must be at least the sum of minimum number of uppercase characters and minimum number of lowercase characters. Disabled by default. Valid values are:
  - 0 through 16 – the minimum number of special characters required for a password.
  - -1 – the password cannot contain special characters.
- `min special char in password` – the minimum number of special characters for a password. Valid values are:
  - 0 through 16 – the minimum number of special characters required for a password.
  - -1 – the password cannot contain special characters.
- `min upper char in password` – the minimum number of uppercase letters for a password. Disabled by default. Valid values are:
  - 0 through 16 – the number of uppercase letters required for a password.
  - -1 – the password cannot contain uppercase characters.
- `min lower char in password` – the minimum number of lowercase letters for a password. Valid values are:
  - 0 through 16 – the number of uppercase letters required for a password.
  - -1 – the password cannot contain uppercase characters.
- `minimum password length` – the minimum password length. You can set a minimum password length from 0 to 30. The value you specify with must be at least the sum of all other minimum requirements. For example, `minimum password length` must be set to at least 10 if you have set:
  - `minimum digits in password` to 3
  - `minimum special characters in password` to 2
  - `minimum uppercase characters in password` to 2
  - `minimum lowercase characters in password` to 3
- `password expiration` – the number of days a password can exist before it expires. You specify this value on a global basis. Disabled by default. Valid values are:
  - 0 – the password will never expire.
  - 1 through 32767 – the number of days the password can exist without expiring.
- `password exp warn interval` – the number of days before a password expires that the password expiration warning messages displays. These messages display with every successful login until the password is changed or it expires. This value must be less than or equal to the password expiration. Disabled by default.
  Valid values are 0 to 365.
- `maximum failed logins` – the maximum number of failed logins that can occur before the login is locked. Specify this value globally. Disabled by default. Valid values are:
  - 0 – logins are never locked, regardless of the number of failed login attempts.
  - 1 through 32767 – the number of failed logins that can occur before the login is locked.

- `expire login` changes the login status to expired when a system security officer creates or resets a login. The login is then required to change the password on the first login. Disabled by default. Valid values are:
  - 0 – new or reset logins will not expire.
  - 1 – new or reset logins expire; you must reset your password at the first login.

See `sp_passwordpolicy` in the *Reference Manual: Procedures*.

4.2.7.2 Password Complexity Option Cross Checks

Some password complexity options have interaction implications.

- `minimum password length` must be at least the sum of `min digits in password`, `min alpha in password`, and `min special characters in password`.
- `min alpha in password` must be at least the sum of `min upper char in password` and `min lower char in password`.
- `systemwide password expiration` must be greater than `password exp warn interval`.

For the purpose of the above cross-checks, if SAP ASE encounters a password complexity option value of -1, it interprets that as a value of 0. If an option is not set, SAP ASE interprets the option value to be 0 as well.

SAP ASE prints warnings for each new password complexity option that fails to satisfy the cross-checks. Option setting, however, is successful.

4.2.7.3 Set Password Complexity Checks

Set the password complexity options at the login level using `create login` or `alter login`. Set the options at the global level using `sp_passwordpolicy` or `sp_configure`.

Password checks and policies for authentication	**Configuration parameters specified using** `sp_configure`	**Password complexity options specified using** `sp_passwordpolicy`	**Per-login overrides specified using** `alter login`
Password expiration	`system-wide password expiration`	`system-wide password expiration`	`password expiration`
Digits in password	`check password for digit`	`min digits in password`	N/A
Alphabetic characters in password	N/A	`min alpha in password`	N/A
Password length	`minimum password length`	`minimum password length`	`min passwd length`

Password checks and policies for authentication	Configuration parameters specified using `sp_configure`	Password complexity options specified using `sp_passwordpolicy`	Per-login overrides specified using `alter login`
Failed logins lockout	`maximum failed logins`	`maximum failed logins`	`max failed attempts`
Disallow simple passwords	N/A	`disallow simple passwords`	N/A
Special characters in password	N/A	`min special char in password`	N/A
Uppercase letters in password	N/A	`min upper char in password`	N/A
Lowercase letters in password	N/A	`min lower char in password`	N/A
Password expiration warning interval	N/A	`password exp warn interval`	N/A
Resetting your password at first login	N/A	`expire login`	N/A
Custom password complexity checks	N/A	N/A	N/A

Because you can set password configuration options on a global and per-login basis, the order of precedence in which the password options is applied is important.

When applying password options, the order of precedence is:

1. Existing per-login parameters
2. Password complexity options
3. Existing global password options

Example: Example 1

Creates a new login and sets the minimum password length for "johnd" to 6:

```
create login johnd with password complex_password min
      password length '6'
```

These global options for login "johnd" create two minimum password length requirements for login "johnd", and sets restrictions about digits in the password:

```
sp_configure 'minimum password length', '8'
sp_configure 'check password for digit', 'true'
```

```
sp_passwordpolicy 'set', 'min digits in password', '2'
```

If you then try to alter the password for login "johnd":

```
alter login johnd with password complex_password modify password 'abcd123'
```

Passwords are checked in the following order:

1. Per-login existing options check: minimum password length must be greater than 6. This is true and the check passes.
2. New options: minimum digits in password must be greater than 2. This is true and the check passes.
3. Existing global options: minimum password length specified here is not checked because there is already a per-login check for the login "johnd".
4. The check password for digit option is redundant because it is already checked when the minimum number of digits is turned on and set to 2.

Once the designated sequence is checked, and the new password for login "johnd" passes these checks, the password is successfully change.

Example: Example 2

If you enter the following for user "johnd", SAP ASE first checks the per-login existing options, and determines the minimum password length is set to 6, but that you have attempted to alter the password to use only 4 characters:

```
alter login johnd with password complex_password modify
      password abcd
```

The check fails, and an error message is printed. Once one password complexity check fails, no additional options are checked.

Example: Example 3

Creates a new login with the following password configuration options and sets the minimum password length for login `johnd` to 4:

```
create login johnd with password complex_password min
      password length 4
```

This is a per-login, existing option. When you add the following, you have created a global requirement that the minimum number of digits for a password must be 1:

```
sp_passwordpolicy 'set', 'min digits in password', '1'
```

If you then attempt to alter the password for login `johnd` as follows:

```
alter login johnd with password complex_password modify
      password abcde
```

These checks are performed in the following order:

1. Per-login existing options check: the minimum password length of a new password is 4. The password "abcde" is greater than 4, so this check passes.
2. New global requirement check: the minimum digits in a password is set to 1, globally. This check fails.

The password is not changed and an error message is printed.

To alter a password, all the checks must pass.

4.2.8 Enable Custom Password Checks

A system security officer can write user-defined stored procedures that enable custom password checks.

To implement password history checks, create a new user table to store password histories:

```
create table pwdhistory
(
      name varchar(30)not null,  -- Login name.
      password varbinary(30)not null,  -- old password.
      pwdate datetime not null,  -- datetime changed.
      changedby varchar(30)not null  -- Who changed.
)
go
```

Create a new stored procedure `master.dbo.sp_extrapwdchecks` which saves previously used passwords in an encrypted form in the pwdhistory table and disallows reuse of used passwords. The `sp_extrapwdchecks` user-defined stored procedure is called automatically when either the `create login` or `alter login … modify password` commands are executed. The following is an example of the implementation of `sp_extrapwdchecks`:

```
create proc sp_extrapwdchecks
(
@caller_password varchar(30) = NULL, -- the current password of caller
@new_password    varchar(30), -- the new password of the target acct
@loginame        varchar(30) = NULL -- user to change password on
)
as
begin
declare @current_time datetime,
        @encrypted_old_pwd varbinary(30),
        @encrypted_new_pwd varbinary(30),
        @salt varchar(8),
        @changedby varchar(30),
        @cutoffdate datetime
        select @changedby = suser_name()
        select @salt = null

        -- NOTE : caller_password and/or loginame arguments can be null.
        -- In these cases, password history checks should be skipped.

        -- @loginame is null when SSO creates a new login account
        -- using "create login" command.

        -- @caller_password is null when

        -- 1. SSO creates a new login account using
        -- "create login" command.
```

```
        -- 2. SSO modifies the login account's password using
        -- "alter login … modify password" command.

        -- Business logic for custom password checks should be
        -- implemented here.
        -- If there is no need to maintain password history, return
        -- from here.
        if (@loginame is NULL)
        begin
            return 0
        end

        -- Change this line according to the needs of your installation.
        -- This checks below keep history of 12 months only.
        select @current_time = getdate(),@cutoffdate = dateadd
           (month, -12, getdate())
        delete master..pwdhistory
        where name = @loginame
        and pwdate < @cutoffdate
        select @salt = substring(password, 1, 8) from master..pwdhistory
            where pwdate =
                (select max(pwdate) from master..pwdhistory where
                name=@loginame)and name=@loginame
        if @salt is null
        begin

            select @salt = substring(hash
                (password_random(8), 'sha1'), 1, 8)
        end
        select @encrypted_new_pwd = @salt + hash
                (@salt + @new_password, 'sha1')
        if not exists ( select 1 from master..pwdhistory
            where name = @loginame and password = @encrypted_new_pwd )
        begin
            -- new password has not been used before
            if (@loginame != @changedby)
            begin
                return 0
            end
            -- Save old password
            select @encrypted_old_pwd = @salt + hash
                (@salt + @caller_password, 'sha1')
            insert master..pwdhistory
                select @loginame,
                    @encrypted_old_pwd,@current_time, @changedby
            return (0)
        end
        else
        begin
               raiserror 22001 --user defined error message
        end
    end
    go
```

Use `sp_addmessage` to add the user-defined message 22001. A `raiserror` 22001 indicates a custom password-complexity check error.

`sp_extrapasswordchecks` must use `raiserror` to signal a failure to the SAP ASE server. Use `sp_addmessage` to add error message for this failure in the SAP ASE server.

> **i Note**
>
> Do not use `raiserror` to get the expected behavior. `raiserror` updates the `@@error` global variable. `@@error` is also updated each time you execute a T-SQL statement, including `print` and `if`. If `raiserror`

is followed by any T-SQL statement, `@@error` gets overwritten, and `sp_extrapwdchecks` fails to return an error for a failed password if `raiserror` is followed by any TSQL statement.

The following user-defined stored procedure (`sp_cleanpwdchecks`) can be used to clean-up the password history using `sp_extrapwdchecks`.

```
create proc sp_cleanpwdchecks
(
            @loginame      varchar(30)
                        -- user to change password on
)
as
begin
delete master..pwdhistory
where name = @loginame
end
      go
```

Once the two procedures above are defined and installed in the `master` database, they are called dynamically during the password complexity checks.

4.2.9 Set the Login and Role Expiration Interval for a Password

System administrators and system security officers can set the login or role expiration when the login or role is created, or change an existing login or role expiration.

Command	Expiration option
`create login`	Specify the expiration interval for a login password at creation.
`alter login`	Change the expiration interval for a login password.
`create role`	Specify the expiration interval for a role password at creation (only the system security officer can issue `create role`).
`alter role`	Change the expiration interval for a role password (only the system security officer can issue `alter role`).

These rules apply to password expiration for logins and roles:

- A password expiration interval assigned to individual login accounts or roles overrides the global password expiration value. This allows you to specify shorter expiration intervals for sensitive accounts or roles, such as system security officer passwords, and more relaxed intervals for less sensitive accounts such as an anonymous login.
- A login or role for which the password has expired is not directly activated.
- The password expires at the time of day when the password was last changed after the number of days specified by `password expiration interval` has passed.

For details on the syntax and rules for the commands and system procedures, see the appropriate *Reference Manual*.

4.2.9.1 Circumvent Password Protection

Circumventing the password-protection mechanism may be necessary in automated login systems. You can create a role that can access other roles without passwords.

A system security officer can bypass the password mechanism for certain users by granting the password-protected role to another role, and grant the password-protected role to one or more users. Activation of this role automatically activates the password-protected role without having to provide a password.

For example:

Jane is the system security officer for ABC Inc., which uses automated login systems. Jane creates the following roles:

- `financial_assistant`

  ```
  create role financial_assistant with passwd "L54K3j"
  ```

- `accounts_officer`

  ```
  create role accounts_officer with passwd "9sF6ae"
  ```

- `chief_financial_officer`

  ```
  create role chief_financial_officer
  ```

Jane grants the roles of `financial_assistant` and `accounts_officer` to the `chief_financial_officer` role:

```
grant role financial_assistant, accounts_officer to chief_financial_officer
```

Jane then grants the `chief_financial_officer` role to Bob:

```
grant role chief_financial_officer to bob
```

Bob logs in to the server and activates the `chief_financial_officer` role:

```
set role chief_financial_officer on
```

The roles of `financial_assistant` and `accounts_officer` are automatically activated without Bob providing a password. Bob can now access everything under the `financial_assistant` and `accounts_officer` roles without having to enter the passwords for those roles.

4.2.9.2 Create a Password Expiration Interval for a New Login

Use `create login` to set the password expiration interval for a new login.

This example creates the new login "joe" with the password "Djdiek3", and sets the password expiration interval for "joe" to 2 days:

```
create login joe with password Djdiek3 password expiration 30
```

The password for “joe” expires after 30 days from the time of day the login account was created, or 30 days from when the password was last changed.

See `create login` in *Reference Manual: Procedures*.

4.2.9.3 Create a Password Expiration Interval for a New Role

Use `create role` to set the password expiration interval for a new role.

This example creates the new role `intern_role` with the password “temp244”, and sets the password expiration interval for `intern_role` to 7 days:

```
create role intern_role with passwd "temp244", passwd expiration 7
```

The password for intern_role expires after 7 days from the time of day you created the role, or 2 days from when the password was last changed.

See `create role` in *Reference Manual: Commands*.

4.2.9.4 Creation Date Added for Passwords

Passwords are stamped with a creation date equal to the upgrade date of a given server.

The creation date for login passwords is stored in the `pwdate` column of `syslogins`. The creation date for role passwords is stored in the `pwdate` column of `syssrvroles`.

4.2.9.5 Change or Remove Password Expiration Interval for Login or Role

Use `alter login` to change the password expiration interval for an existing login, add a password expiration interval to a login that did not have one, or remove a password expiration interval.

`alter login` only effects login passwords, not role passwords.

This example changes the password expiration interval for the login “joe” to 30 days:

```
alter login joe modify password expiration 30
```

The password expires 30 days from the time of day you ran password expiration.

See `alter login` in *Reference Manual: Commands*.

4.2.9.6 Secure Login Passwords on the Network

The use of asymmetric encryption is allowed to securely transmit passwords from client to server using the RSA public key encryption algorithm.

SAP ASE generates the asymmetric key pair and sends the public key to clients that use a login protocol. For example, the client encrypts the user's login password with the public key before sending it to the server. The server decrypts the password with the private key to begin the authentication of the client connecting.

You can configure SAP ASE to require clients to use a login protocol. Set the configuration parameter `net password encryption reqd` to require all user name- and password-based authentication requests to use RSA asymmetric encryption.

The `sp_configure` parameter `net password encryption reqd` supports a value of 3, which indicates the server should only allow incoming clients that are using EPEP login protocol. The values 0, 1, and 2 also allow EPEP login protocol to be used when a client that supports the login protocol attempts to use it with an SAP ASE that implements the EPEP login protocol.

Setting the value to 2 or 3 increases network memory to support the maximum configured connections using this protocol. The `additional network memory` configuration parameter dynamically adds more memory to the network memory pool used by EPEP. When the value of `net password encryption reqd` is set to 3, the KPP Handler goes into sleep status, because there is no need to provide new RSA key pair for every connection. This means that you will avoid spending CPU cycles generating unique RSA keypairs for each connection, resulting in a more efficient CPU utilization.

You can use the `sp_who` command to check the KPP Handler status.

Two versions of the login protocol using RSA asymmetric encryption are supported. One ensures a unique keypair per login session and the second employs a random number during the login protocol. When there are many user connections using network password encryption, the unique keypair per session may cause computation peaks due to computation of new keypairs. The second approach is less computationally demanding. The second approach requires a recompiled client program that supports the newer login protocol that uses RSA asymmetric encryption with a random number.

Generating an Asymmetric Key Pair

For RSA asymmetric encryption with random number, SAP ASE generates a new key pair:

- At each server start-up,
- Automatically at 24-hour intervals using the housekeeper mechanism, and
- When an administrator with sso_role requests key pair regeneration.

The key pair is kept in memory. A message is recorded in the error log and in the audit trail when the key pair is regenerated.

For RSA asymmetric encryption without random number, by default, a key pair is generated for each connection.

Procedure `sp_passwordpolicy` option `unique keypair per session` may be used to turn on or off the generation of a key pair for each connection with this login protocol. However this should only be used in environments where network password security is not a concern because the key pair is reused without the benefit of the random number component.

To generate the key pair on demand, use:

```
sp_passwordpolicy "regenerate keypair"
```

> **i Note**
>
> Depending on the system load, there may be a delay between the time this command is executed and the time the key pair is actually generated. This is because the housekeeper task runs at a low priority and may be delayed by higher priority tasks.

To generate the key pair at a specific time, use:

```
sp_passwordpolicy "regenerate keypair", <datetime>
```

where `<datetime>` is the date and time you want to regenerate the key pair.

For example, a datetime string of "Jan 16, 2007 11:00PM" generates the key pair at the specified time. The datetime string can also just be a time of day, such as "4:07a.m.". When only time of day is specified, key-pair regeneration is scheduled for that time of day in the next 24-hour period.

`sp_passwordpolicy` lets you configure the frequency of key-pair regeneration, as well as what should be done when a key pair generation fails:

- 'keypair regeneration period', { ([`<keypair regeneration frequency>`], datetime of first generation) | (keypair regeneration frequency, [`<datetime of first generation>`]) }
- "keypair error retry [wait | count]", "`<value>`"

See `sp_passwordpolicy` in *Reference Manual: System Procedure*.

Server Option "net password encryption"

also acts as a client when establishing a remote procedure call (RPC).

When connecting to remote servers, the `net password encryption` option is used to determine whether it will use password encryption.

Either RSA or proprietary algorithms are used when this server option is set to `true`. The command to enable `net password encryption` is:

```
sp_serveroption server, "net password encryption", "true"
```

The setting is stored in `master..sysservers` and you can display the value of server options using the `sp_helpserver` stored procedure.

The default value for `net password encryption` is `true` for any new server added using `sp_addserver`. During upgrade, the `net password encryption` is set to `true` for `sysservers` entries with an `ASEnterprise` class value. No other server classes are modified. This improves password security between two communicating SAP ASE servers.

> **i Note**
>
> The administrator can optionally reset `net password encryption` to `false` if you encounter problems establishing a connection to a server. However, if the option is set to false, passwords are transmitted in clear text on the network.

Backward Compatibility

- SAP recommends that you use the RSA algorithm to protect passwords on the network.
- To use the RSA algorithm with random number, you must have SAP ASE version 15.7 ESD #1 or later and use new Connectivity SDK clients version 15.7 ESD #1 or later.
- To use the RSA algorithm, without random number, you must have SAP ASE version 15.0.2 and new Connectivity SDK clients version 15.0 ESD #7 and later.
- SAP provides the `net password encryption reqd` configuration parameter and the `net password encryption` server option to allow settings equivalent to versions earlier than 15.0.2 and maintain backward compatibility with older clients and older servers.
- Older clients that do not support the RSA algorithm can set the property to encrypt passwords using the proprietary algorithm, which has been available since version 12.0.
- New clients that support both RSA and proprietary algorithms can set properties for both algorithms. When communicating with such clients, SAP ASE 15.0.2 and later uses RSA encryption. A pre-15.0.2 SAP ASE uses the proprietary algorithm.

See *Reference Manual: Configuration Parameters > net password encryption required*.

4.2.10 Secure Login Passwords Stored on Disk and in Memory

Login passwords used to authenticate client connections are stored securely on disk as SHA-256 hash digest.

The SHA-256 algorithm is a one-way encryption algorithm. The digest it produces cannot be decrypted, making its storage on disk secure. To authenticate the user connection, the SHA-256 algorithm is applied to the password sent by the client, and the result compared with the value stored on disk.

To prevent dictionary-based attacks on login passwords stored on disk, a salt is mixed with the password before the SHA-256 algorithm is applied. The salt is stored along with the SHA-256 hash, and used during login authentication.

SAP recommends using only SHA-256 as soon as you are certain that there will be no downgrades to an earlier versions. Consider the trade-offs when making this decision; should there be a need to downgrade to a pre-15.0.2 release, it requires administrator intervention to unlock user login passwords.

4.2.11 Character Set Considerations for Passwords

Passwords and other sensitive data that is encrypted must determine the character set of the clear text to accurately interpret the result when it is decrypted, or when hash values are compared during authentication.

For example, a client connects to SAP ASE using `isql` and establishes a new password. Regardless of the character set used in the client, characters are always converted to the server's default character set for processing within SAP ASE. Assuming the default character set is "iso_1," consider the command:

```
alter login loginName with password <oldPasswd> modify password <newPasswd>
```

The password parameters are `varchar`, and are expressed as a quoted string and stored with "iso_1" encoding before encryption. If the default character set changes later, the encrypted password remains an encrypted string of characters encoded with the original default character set. This may result in authentication failure due to mismatched character mapping. Although changing the default character set is a rare occurrence, it becomes more important when migration occurs between platforms.

The clear text password is converted to canonical form before encryption so that the password can be used across platforms, chip architectures, and character sets.

To use canonical form for storage in `syslogins`:

1. Convert the clear text password string to UTF-16.
2. Convert the UTF-16 string to network byte order.
3. Append a small buffer (the salt) with random bytes to the password.
4. Apply the SHA-256 hash algorithm.
5. Store digest, salt, and version in the `password` column.

At authentication time:

1. Convert the clear text password string to UTF-16.
2. Convert the UTF-16 string to network byte order.
3. Append the salt from the `password` column in `syslogins` to the password.
4. Apply the hash algorithm.
5. Compare results with password column in `syslogins`, if they match then authentication is successful.

4.2.12 Upgrade Behavior

Behavior changes are made on the master database when upgrading.

Behavior Changes on an Upgraded master Database

When you upgrade the `master` database, encrypted passwords are maintained in `syslogins` catalogs using algorithms from the earlier- and the upgraded version in the `password` column.

Users can call `sp_displaylogin` to determine which "`Login password encryption`" a login uses.

On first authentication of a login after an upgrade:

- The user authenticates using the contents of the `password` column and the old algorithm.
- The `password` column is updated with the old encryption algorithm followed by the new encryption algorithm.

On subsequent authentication of a login after upgrade, before "`allow password downgrade`" is set to 0, the user authenticates using the new algorithm.

Behavior Changes on a New master Database

In a new `master` database, or in an upgraded `master` database after `allow password downgrade` is set to 0, the server maintains encrypted passwords in `syslogins` using only the new algorithm in the `password` column. Only the SHA-256 algorithm authenticates the connection requests and stores the password on disk.

Issue `sp_passwordpolicy` to determine if a server was upgraded (for example, from version 15.0 to 15.0.2) and maintains passwords using algorithms from the pre- and post-upgraded server, or if the server is newly installed and includes a `master` database that uses the most recent algorithm (from the 15.0.2 version):

```
sp_passwordpolicy 'list', 'allow password downgrade'
```

4.2.13 Login Password Downgrade

To ease the transition to the on-disk encryption algorithm when migrating from versions earlier than 15.0.2, the password policy `allow password downgrade` is available.

After an upgrade from versions earlier than 15.0.2, the policy has a value of 1 to indicate that passwords are stored in both the proprietary algorithm used in earlier versions and the SHA-256 algorithm used in 15.0.2 and later.

As long as passwords are stored in both old and new forms, you can downgrade to 15.0 without resetting user passwords. When the policy `allow password downgrade` is set to `0`, passwords are stored only in SHA-256 form, which is incompatible with older versions. When downgrading to previous releases, only passwords stored in SHA-256 are reset to random passwords and stored in the old form compatible with older versions.

To end the period when password downgrade is allowed, execute:

```
sp_passwordpolicy 'set', 'allow password downgrade', '0'
```

Before executing this command, examine login accounts with `sp_displaylogin` to determine if the login account has been used, and whether the password is stored in SHA-256 encoding. If is not, the account is automatically locked and reset with a generated password. To use the account again, you must unlock the account and give the user a newly generated password.

You may want to save the output from this command because it can contain information about locked login accounts and generated passwords for those accounts.

When the password downgrade period ends:

- The `datetime` when the password downgrade period ended is recorded in `master.dbo.sysattributes`.
- The value of each `password` column in `syslogins` is rewritten to use only the new password on-disk structure.
- The logins that have not transitioned to the new algorithm have the password reset to a new server-generated password in SHA-256 format, and the login is locked. The generated password is displayed only to the administrator executing the `sp_passwordpolicy` procedure above. The lock reason is set to 3 ("Login or role not transitioned to SHA-256").

After the `sp_passwordpolicy` procedure completes:

- Login authentication uses only SHA-256.
- Only the new password on-disk structure for the `password` column is used.
- Attempts to use the locked logins fail authentication. To use the locked logins, you must unlock the login with `sp_locklogin` and the user must use the password generated by `sp_passwordpolicy`. Alternatively, you may prefer to assign a new password instead of the generated password for locked login accounts.

Example 1

This example prepares an upgraded server to use only SHA-256. Examine login accounts to determine which encryption is used by the account using `sp_displaylogin`.

```
1> sp_displaylogin login993
2> go
Suid: 70
Loginame: login933
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Apr 20 2007 2:55PM
Password expiration interval: 0
Password expired: NO
Minimum password length: 0
Maximum failed logins: 3
Current failed login attempts:
Authenticate with: ANY
Login Password Encryption: SYB-PROP
Last login date:
(return status = 0)
```

The value `SYB-PROP` from the line `Login Password Encryption: SYB-PROP` indicates that only the SAP-proprietary encryption is used for this account. This login has not been used before the upgrade to 15.0.2 and later, and will be locked, and its password reset if `sp_passwordpolicy 'set', 'allow password downgrade', '0'` is executed.

After the first login to the account after upgrading to 15.0.2, the line changes to show that both old and new encryption is used:

```
Login Password Encryption: SYB-PROP,SHA-256
```

This is the desired state for all active login accounts, so that executing `sp_passwordpolicy 'set', 'allow password downgrade', '0'` does not lock and reset the password for accounts.

After you execute `sp_passwordpolicy 'set', 'allow password downgrade', '0'`, only SHA-256 encryption is used, and you see:

```
Login Password Encryption: SHA-256
```

Login accounts that show this value are now using the stronger, on-disk encryption algorithm.

When all passwords have been changed to use the new algorithm, re-executing `sp_passwordpolicy` shows no accounts reset or locked:

```
1> sp_passwordpolicy 'set', 'allow password downgrade', '0'
2> go
```

```
Old password encryption algorithm usage eliminated from 0 login accounts,
changes are committed.
(return status = 0)
```

Example 2

In this example, 990 out of 1000 login accounts have transitioned to the SHA-256 algorithm, but 10 accounts are still using SYB-PROP algorithm:

```
1> sp_passwordpolicy 'set', 'allow password downgrade', '0'
2> go
```

```
Old password encryption algorithm found for login name login1000, suid 3,
ver1 =5, ver2 = 0, resetting password to EcJxKmMvOrDsC4
Old password encryption algorithm found for login name login999, suid 4,
ver1 =5, ver2 = 0, resetting password to MdZcUaFpXkFtM1
Old password encryption algorithm found for login name login998, suid 5,
ver1 =5, ver2 = 0, resetting password to ZePiZdSeMqBdE6
Old password encryption algorithm found for login name login997, suid 6,
ver1 =5, ver2 = 0, resetting password to IfWpXvGlBgDgW7
Old password encryption algorithm found for login name login996, suid 7,
ver1 =5, ver2 = 0, resetting password to JhDjYnGcXwObI8
Old password encryption algorithm found for login name login995, suid 8,
ver1 =5, ver2 = 0, resetting password to QaXlRuJlCrFaE6
Old password encryption algorithm found for login name login994, suid 9,
ver1 =5, ver2 = 0, resetting password to HlHcZdRrYcKyB2
Old password encryption algorithm found for login name login993, suid 10,
ver1 =5, ver2 = 0, resetting password to UvMrXoVqKmZvU6
Old password encryption algorithm found for login name login992, suid 11,
ver1 =5, ver2 = 0, resetting password to IxIwZqHxEePbX5
Old password encryption algorithm found for login name login991, suid 12,
ver1 =5, ver2 = 0, resetting password to HxYrPyQbLzPmJ3
Old password encryption algorithm usage eliminated from 10 login accounts,
changes are committed.
(return status = 1)
```

> **i Note**
>
> The login name, `suid`, and generated password appear to the administrator executing the procedure. The output of the command shows all 10 accounts that have not transitioned are reset (and locked).

4.2.14 Retain Password Encryption After Upgrade then Downgrade

If you upgrade SAP ASE, then downgrade to an earlier version, use `sp_downgrade_esd` to retain and use the password encryption functionality.

By default, you can downgrade passwords after an upgrade, until you end the password downgrade period.

4.2.14.1 Add Space Before You Upgrade

Additional space is required in the `master` database, and transaction log. Use `alter database` to add additional space to the master database, and transaction log.

Encryption algorithms and password policies:

- Increase the space required for `syslogins` by about 30%.
- Increase the maximum row length by 135 bytes per login account.
- Decrease the ratio of rows per page from about 16 rows per 2K page to 12 rows per 2K page between SAP ASE versions 15.0.1 and 15.0.2. There is a period of time during the downgrade when the value for `allow password downgrade` is 1 (when both old and new password encryption algorithms are used); the ratio further decreases to about 10 rows per 2K page.
  For example, if version 15.0.1 has 1,000 login accounts, and the data fits into 59 pages, the same number of login accounts may require approximately 19 additional pages in version 15.0.2 on a new `master` database, or 33 additional pages if you upgraded from 15.0.1 (with `allow password downgrade` set to 1).

The transaction log requires additional space for the updated `password` column. When users first log in, about 829 2K pages per 1,000 logins is required, and about 343 pages per 1,000 logins for password changes users make during the upgrade and downgrade. To ensure there is sufficient log space, verify that there is approximately one 2K page of free log space per login before starting the password upgrade or downgrade, and when users first login to SAP ASE version 15.0.2 and later.

4.2.14.2 Downgrade

SAP ASE supports downgrading from version 16.x to earlier versions of 16.x. If you are downgrading to an earlier version, you may need to perform additional actions.

If `allow password downgrade` is 0 or NULL, or if a password has been stored in `syslogins` with only the SHA-256 algorithm, use `sp_displaylogin` on login accounts to determine which algorithm is used, or `sp_downgrade_esd "prepare"` to determine which accounts are reset.

The `prepare` option reports whether the server is ready to be downgraded. If the `prepare` option fails, it reports errors that must be fixed. If a downgrade is performed on the server before the errors are fixed, the downgrade fails. For login passwords, `prepare` reports which passwords are reset during the downgrade.

Run `sp_downgrade_esd "prepare"` to verify whether you should run `sp_downgrade_esd`:

```
sp_downgrade_esd 'prepare','<version>',1
```

```
Checking databases for downgrade readiness.
There are no errors which involve encrypted columns.
Allow password downgrade is set to 0. Login passwords
may be reset, if old encryption version of password is
not present.
Warning: New password encryption algorithm found for
login name user103, suid 103.
Password will be reset during the downgrade phase.
sp_downgrade 'prepare' completed.
(return status = 0)
```

```
drop login probe
```

If the login has user entries in databases, from the master database, drop users from databases, and then drop the login:

```
use master
sp_dropuser 'probe'
```

The `probe` login is re-created when you run `installmaster` on the downgraded server.

Before executing `sp_downgrade_esd`, SAP recommends that you drop statistics for `syslogins,` and `syssrvroles`. Doing this avoids invalid column information, such as the length of password column, in `sysstatistics` from being recorded during the downgrade.

To drop statistics for `syslogins,` and `syssrvroles`, enter:

```
delete statistics master..syslogins
delete statistics master..syssrvroles
```

In this example, the execution of `sp_downgrade_esd` locks, and resets the login password for `user103`. The random password generated by SAP ASE is shown only to the client who executes `sp_downgrade_esd`. The administrator can redirect this output to a file so that these passwords are retained, or the administrator can manually reset them once the downgrade is complete, and the server is restarted.

```
sp_downgrade_esd 'downgrade','<version>',1
```

```
Checking databases for downgrade readiness.
There are no errors which involve encrypted columns.
Allow password downgrade is set to 0. Login passwords may be reset, if old
encryption version of password is not present.
Warning: New password encryption algorithm found for login name user103, suid
103 .
Password is reset during the downgrade phase.
Executing downgrade step 1 [sp_passwordpolicy 'downgrade'] for :
- Database: master (dbid: 1)
New password encryption algorithm found for login name user103, suid 103.
Resetting password to 'ZdSuFpNkBxAbW9'.
Total number of passwords reset during downgrade = 1
[ ... output from other downgrade steps ..]
(return status = 0)
```

Additional messages appear in the error log to identify steps that occurred during `sp_downgrade_esd`:

```
00:00000:00006:2007/05/21 05:35:59.19 server  Starting downgrading ASE.
00:00000:00006:2007/05/21 05:35:59.20 server  Downgrade : Downgrading login
passwords.
00:00000:00006:2007/05/21 05:35:59.22 server  Downgrade : Starting password
downgrade.
00:00000:00006:2007/05/21 05:35:59.23 server  Downgrade : Removed sysattributes
rows.
00:00000:00006:2007/05/21 05:35:59.23 server  Downgrade : Updated 1 passwords.
00:00000:00006:2007/05/21 05:35:59.24 server  Downgrade : Removed columns in
syslogins -
lastlogindate, crdate, locksuid, lockreason, lockdate are removed.
00:00000:00006:2007/05/21 05:35:59.26 server  Downgrade : Truncated password
lengths.
00:00000:00006:2007/05/21 05:35:59.28 server  Downgrade : Successfully completed
password
downgrade.
00:00000:00006:2007/05/21 05:35:59.28 server  Downgrade : Marking stored
procedures to
be recreated from text.
00:00000:00006:2007/05/21 05:36:03.69 server  Downgrade : Dropping Sysoptions
system
table.
00:00000:00006:2007/05/21 05:36:03.81 server  Downgrade : Setting master
database minor
upgrade version.
00:00000:00006:2007/05/21 05:36:03.83 server  Downgrade : Setting user databases
minor
upgrade version.
00:00000:00006:2007/05/21 05:36:03.90 server  ASE downgrade completed.
```

`sp_downgrade_esd` makes catalog changes, and modifies password data. The server must be in single user mode to successfully execute `sp_downgrade_esd`. To start the server in single user mode, and to allow only the System Administrator to log in, use the `-m` command line option to start the server.

After running `sp_downgrade_esd`, shut down the server to avoid new logins or other actions that may modify data or system catalogs. If you restart the server after running `sp_downgrade_esd`, the earlier version shuts down and you are again upgraded to the later version.

4.2.15 Expire Passwords When allow password downgrade is Set to 0

Expire passwords in `syslogins` at the end of the password downgrade period.

To configure login passwords to expire, use:

```
sp_passwordpolicy "expire login passwords"[, "[<loginame> | <wildcard>]"
```

To configure role passwords to expire, use:

```
sp_passwordpolicy "expire role passwords"[, "[<rolename> | <wildcard>]"
```

To configure stale login passwords to expire, use:

```
sp_passwordpolicy "expire stale login passwords", "<datetime>"
```

To configure stale role passwords to expire, use:

```
sp_passwordpolicy "expire stale role passwords", "<datetime>"
```

Passwords that are not changed since the date you set in the `<datetime>` parameter of the `sp_passwordpolicy "expire stale login passwords,"` expire when you execute the command. Users are automatically required to change their passwords after the password downgrade period ends.

You can also lock stale logins or roles; however this requires you to reset the password manually for legitimate users to access their login account again.

4.2.15.1 Show the Current Value of allow password downgrade

Obtain the current value of `allow password downgrade`.

To obtain the value of `allow password downgrade`, enter:

```
sp_passwordpolicy 'list', 'allow password downgrade'
```

The result set includes the current value, and a message indicating its meaning.

If you have upgraded the `master` database, and are maintaining passwords with the old and new encodings, the result is:

```
sp_passwordpolicy 'list', 'allow password downgrade'
go
```

```
value    message
-------- -----------------------------------------------------
       1 Password downgrade is allowed.
(1 row affected)
```

For an upgraded `master` database that only uses new password encryption, the result is:

```
sp_passwordpolicy 'list', 'allow password downgrade'
go
```

```
value    message
-------- -----------------------------------------------------
       0 Last Password downgrade was allowed on <datetime>.
(1 row affected)
```

For a new `master` database on Adaptive Server 15.0.2 that only uses new password encryption, the result is:

```
sp_passwordpolicy 'list', 'allow password downgrade'
go
```

```
value    message
-------- -----------------------------------------------------
   NULL New master database.
(1 row affected)
```

4.2.16 Passwords in a High Availability Environment

Password security impacts configuration of high availability, the behavior of passwords in `syslogins` between primary, and companion servers.

The primary and companion servers must have equivalent `allow password downgrade` values before you configure them for high availability. The `allow password downgrade quorum` attribute checks whether the value of `allow password downgrade` is the same on both primary, and secondary servers.

If `allow password downgrade` on the primary server is 1, and 0 on the secondary server, then the output of `sp_companion` is:

```
1> sp_companion "primary_server",configure
2> go
```

```
Step: Access verified from Server:'secondary_server' to Server:'primary_server'.
Step: Access verified from Server:'primary_server' to Server:'secondary_server'.
Msg 18836, Level 16, State 1:
Server 'secondary_server', Procedure 'sp_companion', Line 392:
Configuration operation 'configure' can not proceed due to Quorum Advisory Check
failure. Please run 'do_advisory' command to find the incompatible attribute
and fix it.
Attribute Name         Attrib Type      Local Value   Remote Value   Advisory
--------------         -----------      -----------   ------------   --------
allow password downg   allow password             0              1          2
(1 row affected)
(return status = 1)
```

A value of 2 in the `Advisory` column indicates that the user cannot proceed with the cluster operation unless the values on both companions match.

`sp_companion do_advisory` also lists the difference in the value of `allow password downgrade` on both servers.

Run `sp_passwordpolicy 'allow password downgrade'` independently on both the primary, and secondary servers to synchronize the value, and to ensure both servers are in the same state.

4.2.16.1 Passwords Updated After Upgrade

Upon the first connection to the primary server after upgrading and configuring for high availability, the user login password synchronizes on both the primary and companion servers with the same on-disk encryption format.

This avoids password reset or locking when the `allow password downgrade` period ends, and passwords are downgraded to an earlier version of SAP ASE. Login passwords continue to be used without being reset or locked by `sp_passwordpolicy`.

After successfully setting up high-availability environment, end the `allow password downgrade` period separately on the primary and companion servers.

4.3 Establish a Password and Login Policy

Several controls are available for setting policies for logins, roles, and passwords for internal authentication.

The system security officer can:

- Specify the maximum allowable number of times an invalid password can be entered for a login or role before that login or role is automatically locked
- Log in after a lost password
- Manually log and unlock logins and roles
- Display login password information
- Specify the `minimum password length` required server-wide, or for a specific login or role
- Check for password complexity of logins
- Enable custom password checks of logins
- Set the password expiration interval
- Consider login password character set
- Lock inactive login accounts
- Use passwords in a high availability environment

4.4 Login Failure

Users must be successfully authenticated before he or she can access data.

If the authentication attempt fails, the following message is returned and the network connection is terminated:

```
isql -U bob -P badpass
Msg 4002, Level 14, State 1:
Server 'ACCOUNTING'
Login failed.
CT-LIBRARY error:
ct_connect(): protocol specific layer: external error:
The attempt to connect to the server failed
```

This message is a generic login failure message that does not tell the connecting user whether the failure resulted from a bad user name or a bad password.

Although the client sees a generic message for a login failure to avoid giving information to a malicious user, the system administrator may find the reason for the failure to be important to help detect intrusion attempts and diagnose user authentication problems.

The reason for the login failure is provided in the `Errornumber.Severity.State` of the `Other Information` section of `sysaudits.extrainfo` column. Login failure audits have event number 45 and `eventmod 2`.

Set the `sp_audit login` parameter to `on` or `fail` to enable auditing for login failure:

```
sp_audit "login", "all", "all", "fail"
sp_audit "login", "all", "all", "on"
```

4.5 Lock Login Accounts and Roles

To prevent a user from logging in to an SAP ASE server, you can either lock or drop an login account.

Locking a login account maintains the `suid` so that it cannot be reused.

Execute `sp_locklogin` to lock login accounts

Audit records with audit event AUD_EVT_LOGIN_LOCKED (112) are generated under the `login_locked` audit option when the login account is locked because login attempts have reached a configured `maximum failed login` value.

> ⚠ Caution
>
> SAP ASE may reuse the server user ID (`suid`) of a dropped login account when the next login account is created. This occurs only when the dropped login holds the highest `suid` in `syslogins`; however, it can compromise accountability if execution of `drop login` is not being audited. Also, it is possible for a user with the reused `suid` to access database objects that were authorized for the old `suid`.

You cannot drop a login when:

- The user is in any database.
- The login is the last remaining user who holds the system security officer or system administrator roles.

The system security officer can lock or drop a login using `sp_locklogin` or `drop login`. If the system procedure is being logged for replication, the system security officer must be in the master database when issuing the command.

4.5.1 Lock and Unlock Logins and Accounts

Use `sp_locklogin` to lock and unlock logins and accounts or to display a list of locked accounts.

You must be a system security officer to use `sp_locklogin`.

A login can be locked when:

- Its password expires, or
- The maximum number of failed login attempts occur, or
- The system security officer manually locks it, or
- Accounts are locked due to account inactivity

The syntax is:

```
sp_locklogin  [ {<login_name> }, { "lock" | "unlock" } ]
```

where:

- `<login_name>` is the name of the account to be locked or unlocked. The login name must be an existing valid account.
- `all` indicates to lock or unlock all login accounts, except those with sa_role.

- `lock` | `unlock` specifies whether the account is to be locked or unlocked.

The system security officer can use sp_locklogin to manually lock or unlock a login. For example:

```
sp_locklogin "joe" , "lock"
sp_locklogin "joe" , "unlock"
```

Information about the lock status of a login is stored in the status column of `syslogins`.

Display Locked Logins

To display a list of all locked logins, use `sp_locklogin` with no parameters.

You can lock an account that is currently logged in, and the user is not locked out of the account until he or she logs out. You can lock the account of a database owner, and a locked account can own objects in databases. In addition, you can use `sp_changedbowner` to specify a locked account as the owner of a database.

There is always at least one unlocked system security officer's account and one unlocked system administrator's account.

Information about the lock status of a login is stored in the `status` column of `syslogins`.

See `sp_locklogin` in *Reference Manual: Procedures*.

4.5.2 Use syslogins to Track if an Account is Locked

`syslogins` includes columns to track information about locked accounts.

`syslogins` includes the `lastlogindate`, `crdate`, `locksuid`, `lockreason`, and `lockdate` columns to support the last login, and locking inactive accounts, letting an account owner or administrator know if an account is locked, when it was locked, who locked it, and the reason why it was locked.

At login creation, the `crdate` column is set to the current time.

If the `enable last login updates` password policy option is set to 1, the `lastlogindate` column is set to the `datetime` of the login, and the previous value of the column is stored in the process status structure of the login session. The update to `syslogins` and the process status structure can occur at each login to SAP ASE. The default value for `enable last login updates` a new `master` database or an upgraded database is 1. To disable this option execute the procedure using administrator privileges:

```
sp_passwordpolicy 'set', 'enable last login updates', '0'
```

`<@@lastlogindate>` is specific to each user login session, and can be used by that session to determine the date and time of the previous login to the account. If the account has not been previously used or if `enable last login updates` is 0, the value of `<@@lastlogindate>` is NULL.

The transaction log does not log updates to `syslogins..lastlogindate`.

Administrators with sso_role can lock login accounts that are inactive for a given number of days, using:

```
sp_locklogin 'all', 'lock', [@except], '<number of inactive days>'
```

This command has no effect if `enable last login updates` is set to 0 or the value of the `lastlogindate` column is NULL. The range of values for `<number of inactive days>` is 1 – 32767 (days).

The `lockreason` column specifies the reason a login was locked. The value of the `lockdate` column is set to the current `datetime`.

When an account is unlocked, columns `lockreason`, `lockdate`, and `locksuid` are reset to NULL.

The `lockdate`, `locksuid`, and `lockreason` columns are set internally by SAP ASE.

Table 5: The Reasons and Values of locksuid

Values for lockreason	Values for locksuid	Description of `lockreason account`
NULL	NULL	Account has not been locked.
0	`suid` of caller of `sp_locklogin`	Account locked by `locksuid` by manually executing `sp_locklogin`.
1	`suid` of caller of `sp_locklogin`	Account locked due to account inactivity, `locksuid` has manually executed `sp_locklogin 'all', 'lock', 'ndays'`.
2	`suid` of attempted login	Account locked due to failed login attempts reaching maximum failed logins.
3	`suid` of caller of `sp_passwordpolicy set, "allow password downgrade", 0`	Account locked by `locksuid` as the password downgrade period has ended, and login or role has not transitioned to SHA-256.
4	NULL	Account locked due to account inactivity.

4.5.3 Track Locked Roles

Accounting information such as when the role was locked, why it was locked, and who locked is stored in `syssrvroles`, and can be useful for role locking accounting.

There are several reasons roles may be locked:

- Entering the wrong role password a specified number of times. 'max failed_logins' option can be associated with roles during their creation or alteration. It specifies the number of failed role activation attempts after which a role is locked.
- Manually locking the role using `alter role`:

```
alter role <rolename> lock
```

These columns are included in `syssrvroles` for lock information:

- `lockdate` – indicates when the role was locked.

- `locksuid` – indicates who locked the role.
- `lockreason` – gives a reason why it was locked. This is in the form of codes:

Values for lockreason	Value for locksuid	Description of lockreason of role
NULL	NULL	Role is not locked
1	suid of caller of `alter role`	Role locked by suid by manually executing `alter role rolename lock`
2	suid of user whose last attempted role activation led to the role getting locked	Role locked due to failed role activation attempts reading `max failed logins`.

4.5.4 Locking and Unlocking Roles

Use `alter role` to lock and unlock roles.

As system security officer, use `alter role` to manually lock or unlock a role.

For example:

```
alter role physician_role lock
alter role physician_role unlock
```

Information about the lock status of a role is stored in the `status` column of `syssrvroles`.

> **i** Note
>
> In high-availability environments, these `syssrvroles` columns are updated on both the primary and secondary servers.

See `alter role` in *Reference Manual: Commands*.

4.5.5 Lock Logins that Own Thresholds

Thresholds are affected by locked user logins.

- As a security measure, threshold stored procedures are executed using the account name and roles of the login that created the procedure.
  - You cannot drop the login of a user who owns a threshold.
  - If you lock the login of a user who owns a threshold, the user cannot execute the stored procedure.
- The last-chance threshold, and thresholds created by the "sa" login are not affected by `sp_locklogin`. If you lock the "sa" login, the last chance threshold and thresholds created or modified by the "sa" user still fire.

4.6 Manage Login Profiles

System security officers can define, alter, and drop login profiles.

This table summarizes the system procedures and commands used to create and manage login profiles.

Task	Required role	Command or procedure	Database
Create login profiles	System security officer	`create login profile`	Master database
Alter login profiles	System security officer	`alter login profile`	Master database
Drop login profiles	System security officer	`drop login profile`	Master database
Return login profile ID	System security officer	`lprofile_id`	Any database
Return login profile name	System security officer	`lprofile_name`	Any database
Display the name login profiles	System security officer	`sp_displaylogin`	Any database
Displays information about login profiles	System security officer	`sp_securityprofile`	Any database

4.6.1 Login Profile Attributes

Login profile attributes are stored in `syslogins`, `sysloginroles` and `master.dbo.sysattributes`.

This table summarizes the attributes of login profiles.

Attribute	Description
`default database`	Default database.
`default language`	Default language.
`login script`	Valid stored procedure. Stored procedures used as a login script through `create login`, `alter login`, `create login profile`, and `alter login profile`, is restricted to 120 characters.

Attribute	Description
`auto activated roles`	Previously granted user-defined roles that are not password-protected that must be automatically activated on login. An error is generated if the role specified is not granted to the login. By default, user-defined roles are not automatically activated on login.
`authenticate with`	Specifies the mechanism used for authenticating the login account. If `authenticate with <authentication mechanism>` is not specified, the value `ANY` will be used for the login account.
`track lastlogin`	Enables last login updates.
`stale period`	Indicates the duration a login account is allowed to remain inactive before it is locked due to inactivity.
`profile id`	Specifies the login profile ID.

4.6.2 Login Profile Attribute Precedence

The attributes of login profiles are associated with login accounts and applied based on their precedence.

The attributes of login profiles are associated with login accounts using the following precedence:

1. Attribute values from a login profile bound to the login
2. Attribute values from a default login profile
3. Values which have been specified using `sp_passwordpolicy` under the following circumstances:
   - A default login profile does not exit
   - A login profile has not been defined and bound to the account
   - The login profile is set to be ignored (the parameter `with login profile ignore` is specified for the command `create login`)
4. The default value for the attribute

4.6.3 Create Login Profiles

The attributes of a large number of login accounts can be managed by defining a login profile as the default for all login accounts, a subset of login accounts, or individual login accounts.

1. A system security officer creates a login profile for login accounts.
2. A system security officer creates a login account for a new user and associates the login profile to the new login account.
3. A system administrator or database owner adds a user to database or assigns a user to a group.
4. A system security officer grants specific roles to the user or to a login profile.
5. A system administrator, database owner, or object owner grants the user or group specific permissions on specific commands and database objects.

This example creates a login profile `mgr_lp`:

```
create login profile mgr_lp
```

See `create login profile` in the *Reference Manual: Commands*.

4.6.4 Create Default Login Profiles

You can defining a login profile as the default profile for all login accounts.

The following example creates a default login profile named `emp_lp`. If another login profile is currently configured as the default login profile, the default property is removed and applied to `emp_lp`:

```
create login profile emp_lp as default
```

See create login profile in the *Reference Manual: Commands*.

4.6.5 Associate Login Profiles with a Login Account

If a login profile is not specified when creating a login account, the default login profile is associated with the new account.

If a default login does not exist, SAP ASE applies password policy attributes specified by `sp_passwordpolicy` or default attributes.

The following example creates the login account `omar_khayyam` with password `rubaiyat` and associates the account with the login profile `emp_lp`:

```
create login khayyam with password rubaiyat login profile emp_lp
```

The following example modifies the login account `omar_khayyam` and associates the account with the login profile `staff_lp`:

```
alter login khayyam modify login profile staff_lp
```

4.6.6 Ignore Login Profiles

The `ignore login profile` clause is used to disable login profiles associated directly or through a default login profile.

Precedence rules are followed for applying the corresponding attributes of the login account.

The following example creates a login account and specifies to ignore any login profiles.

```
create login maryb with password itsAsecur8 login profile ignore
```

4.6.7 Transfer Existing Login Account Values to a New Login Profile

Transfer existing login account values to a new login profile.

This example shows you how to transfer existing login account values to a new login profile. The login profile `sa_lp` is created with `default database`, `default language`, and `authenticate with` attribute values set to the same values of the login account `ravi`.

```
create login profile sa_lp with attributes from ravi
```

4.6.8 Manual Replication of Login Profiles

The profile ID is an attribute that specifies an ID for a new login profile and is used for manual replication of login profiles across SAP ASE.

For example, if the profile `emp_lp` with a profile ID of 25 is to be created on the replicate master, execute the following command:

```
create login profile emp_lp with profile id 25
```

4.6.9 Grant Roles to Login Profiles

Grant roles to login profiles when creating a login profile.

This example creates the login profile `def_lp` and grants the role `access_role` to the login profile.

```
create login profile def_lp
grant role access_role to def_lp
```

Any login bound to `def_lp` will be implicitly granted `access_role`. The system security officer can specify a role granted to a login profile to behave as a default role for the bound logins, that is, the role is automatically activated in the user's session upon login.

If the default role has been granted to the login using a `where` clause to express an activation predicate, the default role is activated only if the activation predicate evaluates to true.

Related Information

Add or Drop Auto Activated Roles [page 69]

4.6.10 Invoke a Login Script

A login script can be specified to be invoked on login through a login profile.

The syntax is:

```
create login profile ravi_lp with login script 'empNew.script'
```

- A login script can be qualified by specifying the database where it resides and the owner name. When not qualified with a database name, the default database takes precedence over the `master` database.
- If the specified login script is not qualified with an owner name, the owner of the login trigger, which is the current login, takes precedence over the database owner where the login trigger resides.
- Stored procedures used as a login script through `alter login`, `create login profile`, and `alter login profile`, is restricted to 120 characters.

If a global login trigger is specified through `sp_logintrigger`, the login script is invoked after the global login trigger.

Related Information

Using Login Triggers [page 216]

4.6.11 Display Login Profile Information

Display information about login profiles such as the name, ID, and binding information.

Display a Login Profile Name

To display the login profile name of a specified login profile ID or login suid, use:

```
<lprofile_name>([{<login profile id> | <login suid>}])
```

System security officer role is required to view the profile name of the specified login ID if it is not the current user's login ID.

This example displays the login profile name of the specified login profile ID:

```
select lprofile_name(3)
-------------
   intern_lr
```

If no parameter is specified the login profile name of the current user is returned. If a login profile is not associated with the specified login account, then the login profile name of the default login profile is returned. The login profile ignore parameter must not be set.

The login profile name can also be displayed using `sp_displaylogin`. If a login profile is not directly associated with the login account and a default login profile exist, the name of the default login profile is displayed.

Display the Login Profile ID

To display the login profile ID of a specified login profile name or login name, use:

```
<lprofile>_id([{<login profile name> | <login name>}])
```

System security officer role is required to view the profile ID of the specified login name if it is not the current user's login name.

This example displays the login profile name of the specified login profile ID:

```
select lprofile_id('intern_lr')
-------------
   3
```

If a login profile is not associated with the specified login account, then the profile ID of the default login profile is returned. The login profile ignore parameter must not be set.

Display Login Profile Binding Information

Use `sp_securityprofile` to display the login profile attributes associated with a login account.

> **i** Note
>
> A non-privileged login account can only display the attributes of a login profile that it is directly associated with, or the attributes of the default login profile. System security officer role is required to see attributes and bindings of all login profiles.

For more syntax information, see `sp_securityprofile`, in *Reference Manual: System Procedures*.

4.6.12 Modify Login Profiles

The `alter login profile` command can be used to add, drop or change attributes of a login profile and their corresponding values.

If the attributes have not been specified, they will be added to the login profile.

This example removes the login script attribute from the login profile `mgr_lp`. If a login script is specified for the default login profile, it will be invoked on login, otherwise no login script will be invoked.

```
alter login profile mgr_lp drop login script
```

See `alter login profile` in the *Reference Manual: Commands* for complete syntax.

Related Information

Login Profile Attributes [page 63]

4.6.12.1 Add or Drop Auto Activated Roles

Previously granted user defined roles that are not password protected can be automatically activated on login.

The following modifies the login profile `mgr_lp` and automatically activates the roles `mgr_role` and `eng_role` when users associated with `mgr_lp` log in.

```
alter login profile mgr_lp add auto activated roles
      mgr_role, eng_role
```

The auto activated roles status of user defined roles granted to login profiles is indicated in the `sysloginroles.status` column. A value of '1' indicates the granted role must be automatically activated on login. Revoking a role will remove its corresponding row in `sysloginroles` and the role will not be automatically activated on login. Roles are automatically granted to a user's login profile as follows:

1. If a default login profile is associated with the account, any auto activated roles specified in the default login profile are applied.
2. If both a login profile that is directly associated with an account and a default login profile exist, only the auto activated roles specified in a login profile associated directly with the account are applied.

4.6.12.2 Change a Login Profile to be the Default Login Profile

The `as [not] default` clause is used to assign or remove a login profile as the default login profile.

The following statement alters the login profile named `emp_lp` as the default login profile.

```
alter login profile emp_lp as default
```

The following statement removes the login profile named `emp_lp` as the default login profile.

```
alter login profile userGroup_lp as not default
```

4.6.13 Drop Login Profiles

The command `drop login profile` removes the login profile if it is not bound to a login account.

Use `drop login profile with override` to forcefully remove a login profile that is bound to a login account. If the login profile is bound with a login account, the login account will be bound to the default login account, if one exist. If the login profile `ignore` clause has been specified, the clause is removed and the default login profile, if it exists, will be associated with the login account.

This example forcefully removes the login profile `eng_lp` even if it is bound to one or more login accounts.

```
drop login profile eng_lp with override
```

4.7 Securely Automate Database Administration Tasks Using Secure User Store

With secure user store feature, you can connect to the SAP ASE server without explicitly entering username and password in the command line interface; instead, you can establish the connection between a client application and the SAP ASE server using a user-generated key that retrieves the connection information (username, password, host, and port number) from the secure store.

Use the `aseuserstore` utility to create and manage the secure store, generate user keys, and add those keys to the secure store. The feature is supported for ADO.NET, ODBC, jConnect, and SAP ASE native client applications.

Each client category (for example, ADO.NET or ODBC) requires its own entry in the secure store, even when the connection parameters are the same.

This password-less connection option lets you automate database administration tasks without compromising data confidentiality. The `aseuserstore` utility is located in:

- (UNIX) `$SYBASE/OCS-16_0/bin`
- (Windows) `%SYBASE%\OCS-16_0\bin`

4.7.1 How Does the aseuserstore Utility Work?

The `aseuserstore` utility provides commands to store client connection information in the secure store.

Use the `aseuserstore` utility commands to insert, list, and delete connection information in the form of user keys in the secure store. Each user key consists of a username and password, and optionally a hostname, port number, and a database name.

Users are encouraged to periodically change the secure store's encryption key using the `aseuserstore` utility's `CHANGEKEY` command. If there are changes to the client connection parameters (change in the port number, for instance), re-run the `aseuserstore` utility for that key with its new key value (that is, the new port number). Its corresponding entry is then updated in the secure store.

> **i Note**
>
> The `aseuserstore` utility stores the secure user store in `$HOME/.ase/<hostname>`, where `<hostname>` is the name of the host where you have executed the command. If you need to override the default hostname, use the utility's -H option.

4.7.2 Using the aseuserstore Utility With Other SAP ASE Utilities

`isql`, `bcp`, `defncopy`, and `optdiag` utilities include the `-k` parameter, which retrieves username, password, and server connection information from the secure store. With the `-k` parameter, you need to provide the key name while using these utilities to connect to the SAP ASE server.

In the following example, user `ralph` creates a secure store key to connect to SAP ASE:

```
aseuserstore -i
Command: set
Key: ralphkey
Connection info: ralph_server 5000
Username: ralph
Password:
```

User `ralph` now uses this secure store key with the `isql` utility to connect to `ralph_server` instead of the standard `-U<username> -P<password> -S<servername>` parameters:

```
isql -k ralphkey
1>
```

4.7.3 Using the aseuserstore Utility With the Client Applications

To use the secure user store feature, client applications use the following properties in their respective connection strings. It retrieves the connection information from the secure store:

- For CT-Library – Connection property `CS_SECSTOREKEY`
- For JDBC/jConnect – Connection property `SECURE_STORE_KEY`
- For ODBC – Connection property `SecStoreKey`
- For ADO.NET – Connection property `SecStoreKey`

> **i Note**
>
> When using the secure store option, username and password are always retrieved from the secure store. If you specify the username and/or password in the connection string also, it results in an error. Other connection parameters are optional, and may be specified in the connection string if they are not set in the secure store. If you specify connection parameters in both the connection string and secure store, their values must match for the connection to succeed.

4.7.4 Operating System Support

The 64-bit versions of the following platforms support the `aseuserstore` utility and the client applications that use the secure user store feature:

- Microsoft Windows x86-64 64-bit
- HP HP-UX Itanium 64-bit
- Linux x86-64 64-bit
- Solaris SPARC 64-bit (SVR4)
- Solaris x86-64 64-bit
- IBM AIX 64-bit

4.8 Add Users to Databases

The database owner or a system administrator can use `sp_adduser` to add a user to a specific database.

The user must already have an SAP ASE login. The syntax is:

```
sp_adduser <loginame> [, <name_in_db >[, <grpname>]]
```

Where:

- `<loginame>` – is the login name of an existing user.
- `<name_in_db>` – specifies a name that is different from the login name by which the user is to be known inside the database.
  Use `<name_in_db>` to accommodate users' preferences. For example, if there are five users named Mary, each must have a different login name. Mary Doe might log in as "maryd", Mary Jones as "maryj", and so on. However, if these users do not use the same databases, each might prefer to be known simply as "mary" inside a particular database.
  If no `<name_in_db>` parameter is given, the name inside the database is the same as `loginame`.

  > **i Note**
  >
  > This capability is different from the alias mechanism described in "Using Aliases in Databases," which maps the identity and permissions of one user to another.

- `<grpname>` – is the name of an existing group in the database. If you do not specify a group name, the user is made a member of the default group "public." Users remain in "public" even if they are a member of another group.

`sp_adduser` adds a row to the `sysusers` system table in the current database. When a user has an entry in the `sysusers` table of a database, he or she:

- Can issue `use <database_name>` to access that database
- Will use that database by default, if the default database parameter was issued as part of `create login`
- Can use `alter login` to make that database the default

This example shows how a database owner can give access permission to “maryh” of the engineering group “eng,” which already exists:

```
sp_adduser maryh, mary, eng
```

This example shows how to give “maryd” access to a database, keeping her name in the database the same as her login name:

```
sp_adduser maryd
```

This example shows how to add “maryj” to the existing “eng” group, keeping her name in the database the same as her login name by using `null` in place of a new user name:

```
sp_adduser maryj, null, eng
```

Users who have access to a database still need permissions to read data, modify data, and use certain commands. These permissions are granted with the `grant` and `revoke` commands.

Related Information

Change Group Membership [page 76]

4.8.1 Add a “guest” User to a Database

Creating a user named “guest” in a database enables any user with an account to access the database as a guest user.

If a user who has not been added to the database as a user or an aliased user issues the `use <database_name>` command, SAP ASE looks for a guest user. If there is one, the user is allowed to access the database, with the permissions of the guest user.

The database owner can use `sp_adduser` to add a guest entry to the `sysusers` table of the database:

```
sp_adduser guest
```

The guest user can be removed with `sp_dropuser`..

If you drop the guest user from the `master` database, server users who have not yet been added to any databases cannot log in to the server.

> **i Note**
>
> Although more than one individual can be a guest user in a database, SAP ASE can still use the user’s server user ID, which is unique within the server, to audit each user’s activity.

4.8.1.1 “guest” User Permissions

When you install SAP ASE, `master..sysusers` contains a guest entry.

The database owner and the owners of database objects can use `grant` and `revoke` to make the privileges of “guest” either more or less restrictive than those of “public.” “guest” inherits the privileges of “public.”

4.8.1.2 “guest” User in User Databases

In user databases, the database owner adds a guest user that permits all Adaptive Server users to use that database, which saves the owner from having to use `sp_adduser` to explicitly name each user as a database user.

You can use the guest mechanism to restrict access to database objects while allowing access to the database.

For example, the owner of the `titles` table can grant `select` permission on `titles` to all database users except “guest” by executing:

```
grant select on titles to public
sp_adduser guest
revoke all on titles from guest
```

4.8.1.3 “guest” User in Installed System Databases

SAP ASE creates the system `tempdb` database and user-created temporary databases with a guest user.

Temporary objects and other objects created in `tempdb` are automatically owned by user “guest.” `sybsystemprocs`, `sybsystemdb`, and `sybsyntax` databases automatically include the “guest” user.

4.8.1.4 “guest” User in pubs2 and pubs3

The “guest” user entry in the sample databases allows new users to follow the examples in the Transact-SQL Users Guide.

The guest is given a wide range of privileges, including:

- `select` permission and data modification permission on all of the user tables
- `execute` permission on all of the procedures
- `create table`, `create view`, `create rule`, `create default`, and `create procedure` permissions

4.8.2 Add Guest Users to the Server

The system security officer can use `create login` to enter a login name and password that visiting users are instructed to use.

Typically, such users are granted restricted permissions. A default database may be assigned.

> ⚠ Caution
>
> A visitor user account is not the same as the "guest" user account. All users of the visitor account have the same server user ID; therefore, you cannot audit individual activity. Each "guest" user has a unique server ID, so you can audit individual activity and maintain individual accountability. SAP recommends that you do not set up a visitor account to be used by more than one user because you cannot maintain individual accountability.

You can use `create login` to add a visitor user account named "guest" to `master..syslogins`. This "guest" user account takes precedence over the system "guest" user account. If you add a visitor user named "guest" with `sp_adduser`, this impacts system databases such as `sybsystemprocs` and `sybsystemdb`, which are designed to work with system "guest" user in them.

4.8.3 Add Remote Users

You can allow users on another SAP ASE server to execute stored procedures on your server by enabling remote access.

Working with the system administrator of the remote server, you can also allow users of your server to execute remote procedure calls to the remote server.

To enable remote procedure calls, you must reconfigure both the local and the remote servers. See "Managing Remote Servers" in the *System Administration Guide, Volume I*.

4.9 Create Groups

Groups let you grant and revoke permissions to more than one user in a single statement, as well as allow you to provide a collective name to a group of users.

Groups are especially useful if you administer an SAP ASE installation that has a large numbers of users.

Create groups before adding users to a database, since `sp_adduser` can assign users to groups as well as add them to the database.

You must have the system administrator or system security officer role, or be the database owner to create a group with `sp_addgroup`. The syntax is:

```
sp_addgroup <grpname>
```

The group name, a required parameter, must adhere to the rules for identifiers. The system administrator, system security officer, or the database owner can use `sp_changegroup` to assign or reassign users to groups.

For example, to set up the Senior Engineering group, use this command while using the database to which you want to add the group:

```
sp_addgroup senioreng
```

`sp_addgroup` adds a row to `sysusers` in the current database. Therefore, each group in a database, as well as each user, has an entry in `sysusers`.

4.9.1 Change Group Membership

A system administrator, system security officer, or the database owner can use `sp_changegroup` to change a user's group affiliation.

Each user can be a member of only one group other than "public," of which all users are always members.

Before you execute `sp_changegroup`:

- The group must exist.
- The user must have access to the current database (must be listed in `sysusers`).

The syntax for `sp_changegroup` is:

```
sp_changegroup <grpname>, <username>
```

For example, to change the user "jim" from his current group to the group "management," use:

```
sp_changegroup management, jim
```

To remove a user from a group without assigning the user to another group, you must change the group affiliation to "public":

```
sp_changegroup "public", jim
```

The name "public" must be in quotes because it is a reserved word. This command reduces Jim's group affiliation to "public" only.

When a user changes from one group to another, the user loses all permissions that he or she had as a result of belonging to the old group, but gains the permissions granted to the new group.

The assignment of users into groups can be changed at any time.

4.9.2 Create Groups and Add Users

The system security officer, the system administrator, or the database administrator creates a group using `sp_addgroup <group_name>`.

You can grant and revoke permissions at the group level. Group permissions are automatically passed to group members. Every database is created with a group named "public" to which all users automatically belong. Add a user to a group using `sp_adduser` and change a user's group with `sp_changegroup`. .

Groups are represented by an entry in the `sysusers` table. You cannot use the same name for creating a group and a user in the database (for example, you cannot have both a group and a user named "shirley").

4.10 Aliases in Databases

The alias mechanism allows you to treat two or more users as the same user inside a database so that they all have the same privileges.

This mechanism is often used so that more than one user can assume the role of database owner. A database owner can use the `setuser` command to impersonate another user in the database. You can also use the alias mechanism to set up a collective user identity.

For example, suppose that several vice presidents want to use a database with identical privileges and ownerships. If you add the login "vp" to the server and the database and have each vice president log in as "vp," there is no way to tell the individual users apart. Instead, alias all the vice presidents, each of whom has his or her own account, to the database user name "vp."

> **i Note**
>
> Although more than one individual can use the alias in a database, you can still maintain individual accountability by auditing the database operations performed by each user.

The collective user identity from using aliases implies set-ownership for database objects. For example, if user "loginA" is aliased to `dbo` in database `db1`, all objects created by "loginA" in `db1` are owned by `dbo`. However, an object's ownership is concretely recorded in terms of the login name and the creator's database user ID. An alias cannot be dropped from a database if he or she concretely owns objects in that database.

> **i Note**
>
> You cannot drop the alias of a login if that login created objects in the database. In most cases, use aliases only for users who do not own tables, procedures, views, or triggers.

Related Information

Auditing [page 334]
Concrete Identification [page 172]

4.10.1 Add Aliases

Use `sp_addalias` to add an alias for a user. Executing `sp_addalias` maps the user name specified by `<loginame>` to the user name specified by `<name_in_db>`.

The syntax is:

```
sp_addalias <loginame>, <name_in_db>
```

where:

- `<loginame>` – is the name of the user who wants an alias in the current database. This user must have an account but cannot be a user in the current database.
- `<name_in_db>` – is the name of the database user to whom the user specified by `<loginame>` is to be linked. The `<name_in_db>` must exist in `sysusers` in the current database.

Adding an alias adds a row to the system table `sysalternates`.

When a user tries to use a database, a check is performed for the user's server user ID number in `sysusers`. If it is not found, then `sysalternates` is checked. If the user's `<suid>` is found there, and it is mapped to a database user's `<suid>`, the first user is treated as the second user while the first user is using the database.

For example, suppose that Mary owns a database. She wants to allow both Jane and Sarah to use the database as if they were its owner. Jane and Sarah have logins but are not authorized to use Mary's database. Mary executes the following commands:

```
sp_addalias jane, dbo
exec sp_addalias sarah, dbo
```

> ⚠ Caution
>
> Users who are aliased to the database owner have all the permissions and can perform all the actions that can be performed by the database owner, with respect to the database in question. A database owner should carefully consider the implications of vesting another user with full access to a database.

4.10.2 Drop Aliases

Use `sp_dropalias` to drop the mapping of an alternate `<suid>` to a user ID.

Dropping the mapping of an alternate `<suid>` to a user ID deletes the relevant row from `sysalternates`.

The syntax is the following, where `<loginame>` is the name of the user specified by `<loginame>` when the name was mapped with `sp_addalias`:

```
sp_dropalias< loginame>
```

After a user's alias is dropped, the user no longer has access to the database.

You cannot drop an alias if the aliased login created any objects or thresholds. Before using `sp_dropalias` to remove an alias that has performed these actions, remove the objects or procedures. If you still need them after dropping the alias, re-create them with a different owner.

4.10.3 Information About Aliases

To display information about aliases, use `sp_helpuser`.

For example, to find the aliases for "dbo," execute:

```
sp_helpuser dbo
```

```
Users_name     ID_in_db    Group_name   Login_name
----------     --------    ----------   ----------
dbo            1           public       sa
(1 row affected)
```

```
Users aliased to user.
Login_name
----------------------
andy
christa
howard
linda
```

4.11 Obtain Information About Users

Obtain information about users, groups, and current usage using these procedures.

Task	Procedure
Report current users and processes	`sp_who`
Display information about login accounts	`sp_displaylogin`
Report users and aliases in a database	`sp_helpuser`
Report groups within a database	`sp_helpgroup`

4.11.1 Report on Users and Processes

Use `sp_who` to report information about current users and processes.

The syntax is:

```
sp_who [<loginame> | "<spid>"]
```

where:

- `<loginame>` – is the user's login name. If you provide a login name, `sp_who` reports information about processes being run by that user.

- <spid> – is the number of a specific process.

For each process run, sp_who reports the security-relevant information for the server process ID, its status, the login name of the process user, the real login name (if <login_name> is an alias), the name of the host computer, the server process ID of a process that is blocking this one (if any), the name of the database, and the command being run.

If you do not provide a login name or <spid>, sp_who reports on processes being run by all users.

The following example shows the security-relevant results from executing sp_who without a parameter:

```
fid spid status loginame origname hostname blk_spid dbname tempdbname cmd block_xloid threadpool
--- ---- ------ -------- -------- -------- -------- ------ ---------- --- ----------- ----------

0   1  running   sa       sa      sunbird   0 pubs2  tempdb  SELECT           0  syb_default_pool
0   2  sleeping  NULL     NULL              0 master tempdb  NETWORK HANDLER  0  syb_default_pool
0   3  sleeping  NULL     NULL              0 master tempdb  MIRROR HANDLER   0  syb_default_pool
0   4  sleeping  NULL     NULL              0 master tempdb  AUDIT PROCESS    0  syb_default_pool
0   5  sleeping  NULL     NULL              0 master tempdb  CHECKPOINT SLEEP 0  syb_default_pool
```

sp_who reports NULL for the <loginame> for all system processes.

4.11.2 Obtain Information about Login Accounts

Use sp_displaylogin to display information about a specified login account—or login names matching a wild-card pattern—including any roles granted, where <loginame> (or the wildcard matching pattern) is the user login name pattern about which you want information.

The syntax is:

```
sp_displaylogin [<loginame> | <wildcard>]
```

If you are not a system security officer or system administrator, you can display information only about your own account. If you are a system security officer or system administrator, you can use the <loginame> | <wildcard> parameter to access information about any account.

sp_displaylogin displays your server user ID, login name, full name, any roles that have been granted to you, date of last password change, default database, default language, whether your account is locked, any auto-login script, password expiration interval, whether password has expired, the login password encryption version used, and the authentication mechanism specified for the login.

sp_displaylogin displays all roles that have been granted to you, so even if you have made a role inactive with the set command, that role appears. For example, this displays the roles for the sa:

```
sp_displaylogin 'mylogin'
```

```
Suid: 121
Loginame: mylogin
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
```

```
        sybase_ts_role (default ON)
Locked: NO
Date of Last Password Change: Aug 10 2006 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: NONE
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

4.11.3 Obtain Information About Database Users

Use `sp_helpuser` to report information about authorized users of the current database.

Where `<name_in_db>` is the user's name in the current database:

```
sp_helpuser [<name_in_db>]
```

If you give a user's name, `sp_helpuser` reports information about that user. If you do not give a name, it reports information about all users.

The following example shows the results of executing `sp_helpuser` without a parameter in the database `pubs2`:

```
sp_helpuser
```

```
Users_name  ID_in_db  Group_name Login_name
----------  --------  ---------- ----------
dbo         1         public     sa
marcy       4         public     marcy
sandy       3         public     sandy
judy        5         public     judy
linda       6         public     linda
anne        2         public     anne
jim         7         senioreng  jim
```

4.11.4 Find User Names and IDs

To find a user's server user ID or login name, use `suser_id` and `suser_name`.

To find	Use	With the argument
Server user ID	`suser_id`	`(["server_user_name"])`
Server user name (login name)	`suser_name`	`([server_user_ID])`

The arguments for these system functions are optional. If you do not provide one, information about the current user is shown.

This example shows how to find the server user ID for the user "sandy:"

```
select suser_id("sandy")
```

```
------
     3
```

This example shows how a system administrator whose login name is "mary" issues the commands without arguments:

```
select suser_name(), suser_id()
```

```
------------------------------ ------
mary                                4
```

To find a user's ID number or name inside a database, use `user_id` and `user_name`.

Table 6: System functions user_id and user_name

To find	Use	With the argument
User ID	`user_id`	`(["db_user_name"])`
User name	`user_name`	`([db_user_ID])`

The arguments for these functions are optional. If you do not provide one, information about the current user is shown. For example:

```
select user_name(10)
```

```
------------------------------------------------------
NULL
(1 row affected)
```

```
select user_name( )
```

```
------------------------------------------------------
dbo
(1 row affected)
```

```
select user_id("joe")
```

```
------------------------------------------------------
NULL
(1 row affected)
```

4.12 Change User Information

Commands and system procedures used to change passwords, default database, default language, full name, or group assignment.

Task	Required role	System procedure	Master database for: alter/create/drop login/ login profile commands
Change your password	User	`alter login`	Any database
Change another user's password	System security officer	`alter login`	Any database
Change authentication mechanism	System security officer	`alter login` `alter login profile`	Any database
Change full name	System security officer	`alter login`	Any database
Change your own full name	User	`alter login`	Any database
Change default language or default database	System security officer	`alter login profile` `alter login`	Any database
Change the group assignment of a user	System administrator, database owner, or system security officer	`sp_changegroup`	User database
Changing a login profile	System security officer	`alter login profile`	Any database
Configuring a login trigger	System security officer	`alter login profile`	Any database

4.12.1 Change Passwords

All users can change their passwords at any time using `alter login`. The system security officer can use `alter login` to change any user's password.

For example, to the password of the login account named ron, enter:

```
alter login ron with password watsMypaswd modify
password 8itsAsecret
```

See `alter login` in *Reference Manual: Commands*.

4.12.1.1 Require New Passwords

You may choose to use the `systemwide password expiration` configuration parameter to establish a password expiration interval, which forces all users to change their passwords on a regular basis.

Even if you do not use `systemwide` password expiration, it is important, for security reasons, that users change their passwords periodically. See *Reference Manual: Configuration Parameters*..

The configuration parameter is superseded by the password policy settings.

`password expiration interval` specifies the password expiration interval in days. It can be any value between 0 and 32767, inclusive. For example, if you create a new login on August 1, 2012 at 10:30 a.m., with a password expiration interval of 30 days, the password expires on August 31, 2012 at 10:30 a.m.

The column `pwdate` in the `syslogins` table records the date of the last password change. The following query selects all login names whose passwords have not changed since September 15, 2012:

```
select name, pwdate
from syslogins
where pwdate < "Sep 15 2012"
```

4.12.1.2 Null Passwords

Secure login accounts with a non-null password.

Do not assign a null password. When SAP ASE is installed, the default “sa” account has a null password. This example shows how to change a null password to a valid one:

```
alter login sa with password null modify password 8M4LNC
```

> **i** Note
>
> Do not enclose “null” in quotes in the statement.

4.12.1.3 Log in After Lost Password

Use the `dataserver` command to set new passwords sa and sso accounts and roles that are locked or when passwords have been lost.

You can use `dataserver -p<login_name>` if your site encounters any of these situations:

- All system administrator login accounts are locked.
- All system security officer login accounts are locked.
- The password for `sa_role` or `sso_role` has been lost.

The `dataserver` parameter, with the `-p` parameter allows you to set a new password for these accounts and roles. `<login_name>` is the name of the user or the name of the role (`sa_role` or `sso_role`) for which the password must be reset.

When you start with the -p parameter, SAP ASE generates, displays, and encrypts a random password and saves it in master..syslogins or in master..syssrvroles as that account or role's new password.

Sybase strongly recommends that you change the password when the server restarts. For example, to reset the password for user rsmith who has sa_role:

```
dataserver -prsmith
```

To reset the password of the sso_role:

```
dataserver -psso_role
```

4.12.2 Change User Session Information

The set command includes options that allow you to assign each client an individual name, host name, and application name.

This is useful for differentiating among clients in a system where many clients connect to an SAP ASE server using the same name, host name, or application name.

The partial syntax for the set command is:

```
set [clientname <client_name> | clienthostname <host_name> | clientapplname
<application_name>]
```

where:

- <client_name> – is the name you are assigning the client.
- <host_name> – is the name of the host from which the client is connecting.
- <application_name> – is the application that is connecting to SAP ASE.

These parameters are stored in the clientname, clienthostname, and clientapplname columns of the sysprocesses table.

For example, if a user logs in as "client1," you can assign them an individual client name, host name, and application name using commands similar to:

```
set clientname 'alison'
set clienthostname 'money1'
set clientapplname 'webserver2'
```

This user now appears in the sysprocesses table as user "alison" logging in from host "money1" and using the "webserver2" application. However, although the new names appear in sysprocesses, they are not used for permission checks, and sp_who still shows the client connection as belonging to the original login (in the case above, client1). set clientname does not perform the same function as set proxy, which allows you to assume the permissions, login name, and <suid> of another user.

You can set a client name, host name, or application name for only your current client session (although you can view the connection information for any client connection). Also, this information is lost when a user logs out. These parameters must be reassigned each time a user logs in. For example, the user "alison" cannot set the client name, host name, or application name for any other client connection.

Use the client's system process ID to view their connection information. For example, if the user "alison" described above connects with a `spid` of 13, issue the following command to view all the connection information for this user:

```
select * from sysprocesses where spid = 13
```

To view the connection information for the current client connection (for example, if the user "alison" wanted to view her own connection information), enter:

```
select * from sysprocesses where spid = @@spid
```

4.13 Drop Users and Groups

A system administrator, system security officer, or database owner can use `sp_dropuser` or `sp_dropgroup` to drop users and groups from databases.

Dropping Users

A database owner, system security officer, or a system administrator can use `sp_dropuser` to deny a user access to the database in which `sp_dropuser` is executed. (If a "guest" user is defined in that database, the user can still access that database as "guest.")

The following is the syntax, where `<name_in_db>` is usually the login name, unless another name has been assigned with `sp_adduser`:

```
sp_dropuser <name_in_db>
```

You cannot drop a user who owns objects. For tables owned by the user, you can use `alter table modify owner` to change the ownerships to other users before you drop the user. For other types of objects, you cannot transfer ownership of objects, you must drop objects owned by a user before you drop the user. To deny access to a user who owns objects, use `sp_locklogin` to lock his or her account.

You also cannot drop a user who has granted permissions to other users. Use `revoke with cascade` to revoke permissions from all users who were granted permissions by the user to be dropped, then drop the user. You must then grant permissions to the users again, if appropriate.

Dropping Groups

The system security officer, the system administrator, or the database administrator uses `sp_dropgroup` to drop a group. The syntax is:

```
sp_dropgroup <grpname>
```

You cannot drop a group that has members. If you try to do so, the error report displays a list of the members of the group you are attempting to drop.

Related Information

Change Group Membership [page 76]

4.14 Monitor License Use

The License Use Monitor allows a system administrator to monitor the number of user licenses used in SAP ASE, and to securely manage the license agreement data.

You can ensure that the number of licenses used on your SAP ASE server does not exceed the number specified in your license agreement. The License Use Monitor tracks the number of licenses issued; it does not enforce the license agreement. If the License Use Monitor reports that you are using more user licenses than specified in your license agreement, see your SAP sales representative.

You must have system administrator privileges to configure the License Use Monitor; by default the monitor is turned off when SAP ASE is installed or upgraded.

4.14.1 How Licenses are Counted

A license is the combination of a host computer name and a user name. If a user logs in multiple times from the same host machine, one license is used. However, if the user logs in once from host A, and once from host B, two licenses are used.

If multiple users log in from the same host, but with different user names, each distinct combination of user name and host name uses one license.

4.14.2 Configure the License Use Monitor

Use `sp_configure` to specify the number of licenses in your license agreement.

Where `<number>` is the number of licenses:

```
sp_configure "license information" , <number>
```

This example sets the maximum number of user licenses to 300, and reports an overuse for license number 301:

```
sp_configure "license information", 300
```

If you increase the number of user licenses, you must also change the `license information` configuration parameter.

4.14.3 Monitor License Use with the Housekeeper Task

After you configure the License Use Monitor, the housekeeper task determines how many user licenses are in use, based on the user ID and the host name of each user logged in to SAP ASE.

The License Use Monitor updates a variable that tracks the maximum number of user licenses in use:

- If the number of licenses in use is the same or has decreased since the previous housekeeper run, the License Use Monitor does nothing.
- If the number of licenses in use has increased since the previous housekeeper run, the License Use Monitor sets this number as the maximum number of licenses in use.
- If the number of licenses in use is greater than the number allowed by the license agreement, the License Use Monitor issues this message to the error log:

  ```
  Exceeded license usage limit. Contact SAP Sales for additional licenses.
  ```

The housekeeper chores task runs during idle cycles. Both the `housekeeper free write percent` and the `license information` configuration parameter must be set to values greater than or equal to 1 for the License Use Monitor to track license use.

For more information about the housekeeper chores task, see "Using Engines and CPUs," in the *Performance and Tuning Series:Basics*.

4.14.4 Log the Number of User Licenses

The `syblicenseslog` system table is created in the `master` database when you install or upgrade SAP ASE. The License Use Monitor updates the columns in `syblicenseslog` at the end of each 24-hour period.

These are the columns in the `syblicenseslog` table.

Column	Description
`status`	-1 – housekeeper cannot monitor licenses. 0 – number of licenses not exceeded. 1 – number of licensees exceeded.
`logtime`	Date and time the log information was inserted.
`maxlicenses`	Maximum number of licenses used during the previous 24 hours.

`syblicenseslog` looks similar to this:

```
status logdate                   maxlicenses
------ ------------------------- -----------
     0    Jul 17 2012 11:43AM         123
     0    Jul 18 2012 11:47AM         147
```

```
      1    Jul 19 2012 11:51AM          154
      0    Jul 20 2012 11:55AM          142
      0    Jul 21 2012 11:58AM          138
      0    Jul 21 2012  3:14PM          133
```

In this example, the number of user licenses used exceeded the limit on July 19, 2012.

If SAP ASE is shut down, License Use Monitor updates `syblicenseslog` with the current maximum number of licenses used. SAP ASE starts a new 24-hour monitoring period when it is restarted.

The second row for July 21, 2012 was caused by a shutdown and restart of the server.

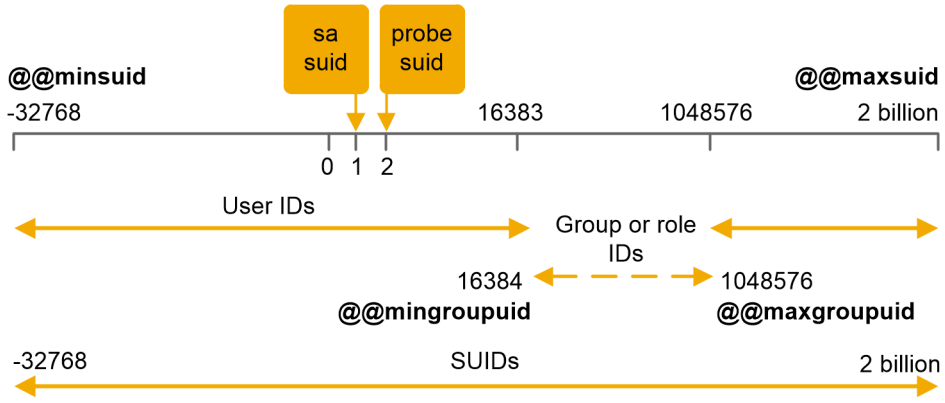
4.15 Number of User and Login IDs

SAP ASE supports over 2,000,000,000 logins per server and users per database. Negative numbers as well as positive numbers are used to increase the range of possible numbers available for IDs.

These are the valid ranges for the ID types:

ID type	Server limits
Logins per server (`<suid>`)	2 billion plus 32K
Users per database (`<uid>`)	2 billion less 1032193
Groups or roles per database (`<gid>`)	16,384 to 1,048,576

Users, groups, and logins available:



You may use negative values for user IDs (`<uid>`).

The server user ID (`<suid>`) associated with a group or a role in `sysusers` is not equal to the negation of their user ID (`<uid>`). Every `<suid>` associated with a group or a role in `sysusers` is set to -2 (INVALID_SUID).

4.16 Login Connection Limitations

Although you are allowed to define more than two billion logins per server, the actual number of users that can connect at one time is limited.

The actual number of users is limited by the:

- Value of the `number of user connections` configuration parameter, and
- Number of file descriptors available. Each login uses one file descriptor for the connection.

> **i** Note
>
> The maximum number of concurrent tasks running on the server is 32,000.

Allowing the Maximum Number of Logins and Simultaneous Connections

Configure the operating system on which SAP ASE is running for at least 32,000 file descriptors, then the value of `number of user connections` to at least 32,000.

> **i** Note
>
> To have more than 64K logins and simultaneous connections, you must first configure the operating system for more than 64K file descriptors. See your operating system documentation for information about increasing the number of file descriptors.

These are the global variables for logins, users, and groups:

Name of variable	What it displays	Value
`<@@invaliduserid>`	Invalid user ID	-1
`<@@minuserid>`	Lowest user ID	-32768
`<@@guestuserid>`	Guest user ID	2
`<@@mingroupid>`	Lowest group or role user ID	16384
`<@@maxgroupid>`	Highest group or role user ID	1048576
`<@@maxuserid>`	Highest user ID	2147483647
`<@@minsuid>`	Lowest server user ID	-32768
`<@@probesuid>`	Probe server user ID	2
`<@@maxsuid>`	Highest server user ID	2147483647

To issue a global variable, enter:

```
select <variable_name>
```

For example:

```
select @@minuserid
-----------
-32768
```

4.17 Obtain Information About Usage: Chargeback Accounting

When a user logs in, the server begins accumulating CPU and I/O usage for that user. Total usage is reported for an individual, or for all users.

Information for each user is stored in the `syslogins` system table in the `master` database.

Reporting Current Usage Statistics

The system administrator can use `sp_reportstats` or `sp_clearstats` to get or clear current total usage data for individuals or for all users.

Displaying Current Accounting Totals

`sp_reportstats` displays current accounting totals for users. It reports total CPU and total I/O, as well as the percentage of those resources used. It does not record statistics for the "sa" login (processes with an `<suid>` of 1), checkpoint, network, and mirror handlers.

Initiating a New Accounting Interval

CPU and I/O statistics are accumulated until you clear the totals from `syslogins` by running `sp_clearstats`. `sp_clearstats` initiates a new accounting interval for users and executes `sp_reportstats` to print out statistics for the previous period.

Choose the length of your accounting interval by deciding how to use the statistics at your site. For example, to do monthly cross-department charging for the percentage of CPU and I/O usage, run `sp_clearstats` once a month.

For detailed information about these stored procedures, see the *Reference Manual: Procedures*.

4.17.1 Specify the Interval for Adding Accounting Statistics

A system administrator can use configuration parameters to decide how often accounting statistics are added to `syslogins`.

To specify how many machine clock ticks accumulate before accounting statistics are added to `syslogins`, use the `cpu accounting flush interval` configuration parameter. The default value is 200. For example:

```
sp_configure "cpu accounting flush interval", 600
```

To find out how many microseconds a tick is on your system, run the following query:

```
 select @@timeticks
```

To specify how many read or write I/Os accumulate before the information is added (flushed) to `syslogins`, use the `i/o accounting flush interval` configuration parameter. The default value is 1000. For example:

```
sp_configure "i/o accounting flush interval", 2000
```

I/O and CPU statistics are flushed when a user accumulates more I/O or CPU usage than the specified value. The information is also flushed when the user exits a session.

The minimum value allowed for either configuration parameter is 1. The maximum value allowed is 2,147,483,647.

5 Manage Roles

Roles are collections of privileges that allow the role assignee to perform their job. The roles supported by SAP ASE let you enforce individual accountability.

SAP ASE provides system roles, such as system administrator and system security officer, and user-defined roles, which are created and granted to users, login profiles, or other roles by a system security officer. Object owners can grant database access as appropriate to a role.

The final steps in adding database users are assigning them special roles, as required, and granting permissions.

> **i** Note
>
> Permission requirements for operations mentioned in this chapter assume that granular permissions is disabled. Operations may differ when granular permissions is enabled.

Related Information

Granular Permissions [page 241]
Manage User Permissions [page 167]

5.1 System-Defined Roles Overview

System roles, values, and tasks.

These are the system roles, the value to use for the `role_name` option of the `grant role` or `revoke role` command, and the tasks usually performed by a person with that role.

Role	Value for *role_granted*	Description
System administrator	`sa_role`	Manage and maintain databases and disk storage
System security officer	`sso_role`	Perform security-related tasks
Operator	`oper_role`	Back up and load databases server-wide
Technical Support	`sybase_ts_role`	Analysis and repair of database structures
Replication	`replication_role`	Replicate user data

Role	Value for *role_granted*	Description
Distributed transaction manager	`dtm_tm_role`	Coordinate transactions across servers
High availability	`ha_role`	Administer and execute failover
Monitor and diagnosis	`mon_role`	Administer and execute performance and diagnostic monitoring
Job Scheduler administration	`js_admin_role`	Administer Job Scheduler
Job Scheduler user	`js_user_role`, `js_client_role`	Create and run jobs through Job Scheduler
Real-time messaging	`messaging_role`	Administer and executer real-time messaging
Web Services	`webservices_role`	Administer Web services
Key custodian	`keycustodian_role`	Create and manage encryption keys
Navigation Server	`navigator_role`	Though navigator_role is present in syssrvroles, this role is not used in SAP ASE and is obsolete.

> **i** Note
>
> sa_role is grantable by a user who has sa_role. All other system role are grantable by a user with sso_role. If a user defined role has been granted both sa_role and other system roles, that role may be granted only by a user who has both sa_role and sso_role.
>
> sa_serverprivs_role is a user defined role that is enabled automatically for the system administrator.

Related Information

Granular Permissions Adds the sa_serverprivs_role [page 255]

5.2 System Roles

System-defined roles restrict system access to authorized users.

System Administrator Privileges

System administrators handle tasks that are not application-specific and work outside the SAP ASE discretionary access control system.

The role of system administrator is usually granted to individual logins. All actions taken by that user can be traced to his or her individual server user ID. If the server administration tasks at your site are performed by a single individual, you may instead choose to use the "sa" account that is installed with SAP ASE. At installation, the "sa" account user can assume the system administrator, system security officer, and operator roles. Any user who knows the "sa" password can log in to that account and assume any or all of these roles.

Having a system administrator operate outside the protection system serves as a safety precaution. For example, if the database owner accidentally deletes all the entries in the `sysusers` table, the system administrator can restore the table (as long as backups exist). There are several commands that can be issued only by a system administrator. They include `disk init`, `disk refit`, `disk reinit`, `shutdown`, `kill`, `disk mirror` , `mount`, `unmount` and several monitoring commands.

In granting permissions, a system administrator is treated as the object owner. If a system administrator grants permission on another user's object, the owner's name appears as the grantor in `sysprotects` and in `sp_helprotect` output.

System administrators automatically assume the identity of a database owner when they log in to a database, and assume all database owner privileges. This automatic mapping occurs, regardless of any aliases assigned to the user. The system administrator can perform tasks usually reserved for the database owner such as `dbcc` commands, diagnostic functions, reading data pages, and recovering data, or indexes.

System Security Officer Privileges

System security officers perform security-sensitive tasks, including:

- Granting the system security officer, operator, and key custodian roles
- Administering the audit system
- Changing passwords
- Adding new logins
- Dropping logins
- Locking and unlocking login accounts
- Creating and granting user-defined roles
- Administering network-based security
- Granting permission to use the `set proxy` or `set session authorization` commands
- Creating login profiles
- Managing encryption

The system security officer can access any database—to enable auditing —but, in general, has no special permissions on database objects (except for encryption keys and decrypt permission on encrypted columns. See *Database Encryption*). An exception is the `sybsecurity` database, where only a system security officer can access the `sysaudits` table. There are also several system procedures that can be executed only by a system security officer.

System security officers can repair any changes inadvertently done to the protection system by a user. For example, if a database owner forgets the password, a system security officer can change the password to allow the database owner to log in.

The system security officers share login management responsibilities with system administrators. System security officers are responsible for managing logins and login profiles.

System security officers can grant all system roles except sa_role. They can also create and grant user-defined roles to users, other roles, login profiles, or groups.

Operator Privileges

Users who have been granted the operator role can back up and restore databases on a server-wide basis without having to be the owner of each database. The operator role allows a user to use these commands on any database:

- `dump database`
- `dump transaction`
- `load database`
- `load transaction`
- `checkpoint`
- `online database`

Technical Support

A Technical Support engineer can use the Technical Support role to display internal memory and on-disk data structures using trace output, consistency checking, and patching data structures.

This role is used for analyzing problems and manually recovering data. Some actions necessary for resolving these issues may require additional system roles for access. SAP recommends that the system security officer grant this role to a knowledgeable SAP engineer only while this analysis or repair is being done.

Replication Role

The user maintaining Replication Server and ASE Replicator requires the replication role. See the *Replication Server Administration Guide* and the *ASE Replicator Users Guide* for information about this role.

Distributed Transaction Manager Role

The distributed transaction manager (DTM) transaction coordinator uses this role to allow system stored procedures to administer transactions across servers. Clients using the DTM XA interface require this role. See *Using Adaptive Server Distributed Transaction Management Features*

High Availability Role

You must have the high availability role to configure the high availability subsystem to administer primary and companion servers through commands and stored procedures. See *Using Sybase Failover in a High Availability System*

Monitoring and Diagnosis

This role is required to administer the monitoring tables. You must have this role to execute a monitoring table remote procedure call and to administer the collection of monitored data. See the *Performance and Tuning Series: Monitoring Tables*

Job Scheduler Roles

The Job Scheduler has three system roles to manage permissions for its operation:

- `js_admin_role` – required to administer Job Scheduler, and provides access to the stored procedures and allow you to modify, delete, and perform Job Scheduler administrative operations.
- `js_user_role` – required for a user to create, modify, delete, and run scheduled jobs using the Job Scheduler stored procedures.
- `js_client_role` – allows users to work with predefined jobs but not to create or alter jobs.

See the *Job Scheduler Users Guide* for more information.

Real-time Messaging Role

Used by the real-time messaging subsystem (RTMS) execute `msgsend`, `msgrecv`, and certain `sp_msgadmin` commands. See the *Messaging Services User's Guide* for more information.

Web Services Role

Used by the Web services subsystem to execute `create service`, `create existing service`, `drop service`, and `alter service` commands. See the *Web Services Users Guide*.

Key Custodian Role

The key custodian role is responsible for key management: creating and altering encryption keys, setting up the system encryption password, setting up key copies for users, and so on. See *Database Encryption*.

5.3 Plan User-Defined Roles

Plan responsibility and positions before implementing roles.

Before you implement user-defined roles, decide:

- The roles you want to create
- The responsibilities for each role
- The position of each in the role hierarchy
- Which roles in the hierarchy are mutually exclusive and if so, at the membership or activation level

Avoid name conflicts when you create user-defined roles by following a naming convention. For example, you can use the "_role" suffix for role names. SAP ASE does not check for such restrictions.

The names of user-defined roles granted directly to users or to login profiles cannot duplicate the name of any login or login profile. If a role must have the same name as a user, avoid conflict by creating a new role, having it contain the original role, and then granting the new role to the user.

After you have planned the roles to create and the relationships among them, decide how to allocate roles according to business requirements and the responsibilities of your users.

The maximum number of roles that a user can activate per user session is 127.

The maximum number of user-defined roles that can be created server-wide is 992.

> **i** Note
>
> The sa_serverprivs_role is a user defined role that is enabled automatically on login for system administrator.

5.4 Create a User-Defined Role

A user with sso_role uses the `create role` command to create a role.

The create role command can only be used in the master database. If a password is used, any user activating the role must specify the password. Roles with passwords cannot be used if the role is to be activated during login as the login's default role or as an automatically activated role granted to a login profile.

For example, to create the `intern_role` without a password, enter:

```
create role intern_role
```

To create the `doctor_role` and assign the password "physician", enter:

```
create role <doctor_role> with passwd "<physician>"
```

Only the system security officer can create user-defined roles.

See `create role` in the *Reference Manual: Commands*.

5.5 Add or Remove Passwords from a Role

Only a system security officer can add or drop a password from a role.

Use the `alter role` command to add or drop a password from either a system or user-defined role:

```
alter role <role_name >
[add passwd <password> | drop passwd]
```

For example, to require the password "oper8x" for the `oper_role`, enter:

```
alter role oper_role add passwd oper8x
```

To drop the password from the role, enter:

```
alter role oper_role drop passwd
```

> **i** Note
>
> When you assign a password to a role, any user granted the role must specify the password when activating the role.

5.6 Role Hierarchies and Mutual Exclusivity

A system security officer can define role hierarchies such that if a user has one role, the user also has roles lower in the hierarchy.

When you grant a role, role1, to another role, say, role2, you set up a hierarchy where role2 contains role1. For example, the "chief_financial_officer" role might contain both the "financial_analyst" and the "salary_administrator" roles.

The chief financial officer can perform all tasks and see all data that can be viewed by salary administrators and financial analysts.

Additionally, you can define a role's mutual exclusivity to enforce static or dynamic separation of duty policies. Roles can be defined to be mutually exclusive for:

- Membership – one user cannot be granted two different roles. For example, you might not want the "payment_requestor" and "payment_approver" roles to be granted to the same user.
- Activation – one user cannot activate, or enable, two different roles. For example, a user might be granted both the "senior_auditor" and the "equipment_buyer" roles, but not permitted to have both roles enabled at the same time.

System roles, as well as user-defined roles, can be defined to be in a role hierarchy, or to be mutually exclusive. For example, you might want a "super_user" role to contain the system administrator, operator, and Technical Support roles. To enforce a separation of roles, you may want to define the system administrator and system security officer roles to be mutually exclusive for membership; that is, one user cannot be granted both roles.

5.6.1 Define and Change Mutual Exclusivity of Roles

To define mutual exclusivity between two roles, use the `alter role` command.

The syntax is:

```
alter role <role1> { add | drop } exclusive { membership | activation } <role2>
```

For example, to define `intern_role` and `specialist_role` as mutually exclusive at the membership level, enter:

```
alter role <intern_role> add exclusive membership< specialist_role>
```

The example above restricts users who have membership in `intern_role` from also being members of `specialist_role`.

To define the `sso_role` and `sa_role` as mutually exclusive at the activation level, enter the following command, which prohibits a user who is a member of sso_role and sa_role from assuming both roles simultaneously:

```
alter role sso_role add exclusive activation sa_role
```

5.6.2 Define and Change a Role Hierarchy

Defining a role hierarchy involves choosing the type of hierarchy and the roles, then implementing the hierarchy by granting roles to other roles.

For example:

```
grant role intern_role to specialist_role
grant role doctor_role to specialist_role
```

This grants to “specialist” all the privileges of both “doctor” and “intern.”

To establish a hierarchy with a “super_user” role containing the `sa_role` and `oper_role` system roles, specify:

```
grant role sa_role to super_user
grant role oper_role to super_user
```

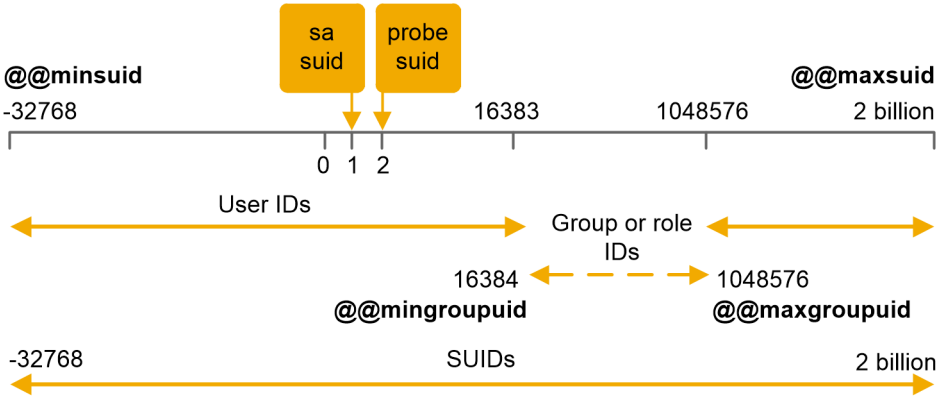
> **i Note**
>
> If a role requires a password to be contained within another role, the user with the role that contains the other does not need to use the password for the contained role. In the example above, assume that the “doctor” role usually requires a password. The user who has the “specialist” role does not need to enter the “doctor” password because “doctor” is contained within “specialist.” Role passwords are only required for the highest level role.

When creating role hierarchies:

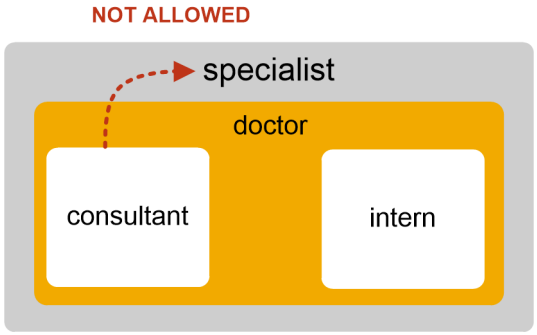
- You cannot grant a role to another role that directly contains it. This prevents duplication.

In the example above, you cannot grant “doctor” to “specialist” because “specialist” already contains “doctor.”

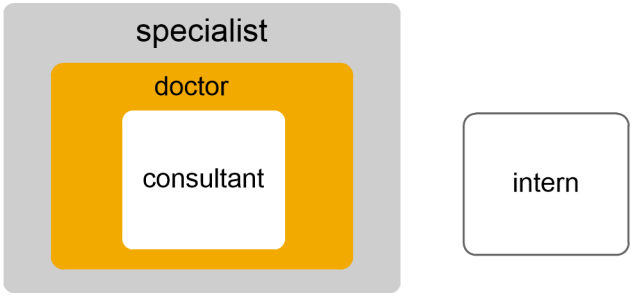
- You can grant a role to another role that does not directly contain it.
  For example, in the Explicitly and implicitly granted privileges figure, you can grant the “intern” role to the “specialist” role, even though “specialist” already contains the “doctor” role, which contains “intern.” If you subsequently dropped “doctor” from “specialist,” then “specialist” still contains “intern.”
  In the Explicitly and implicitly granted privileges figure, “doctor” has “consultant” role permissions because “consultant” has been granted to “doctor.” The “specialist” role also has “consultant” role permissions because “specialist” contains the “doctor” role, which in turn contains the “consultant.”
  However, “intern” does not have “consultant” role privileges, because “intern” does not contain the “consultant” role, either directly or indirectly.



- You cannot grant a role to another role that is contained by the first role. This prevents “loops” within the hierarchy.
  For example, in the Granting a role to a role contained by grantor figure, you cannot grant the “specialist” role to the “consultant” role; “consultant” is already contained in “specialist.”



- When the system security officer grants to a user a role that contains other roles, the user implicitly gets membership in all roles contained by the granted role. However, a role can be activated or deactivated directly only if the user has explicit membership in that role.
- The system security officer cannot grant one role to another role that is explicitly or implicitly mutually exclusive at the membership level with the first role.
  For example, in Mutual exclusivity at membership figure, if the “intern” role is defined as mutually exclusive at the membership level with the “consultant” role, the system security officer cannot grant “intern” to the “doctor.”



- The user can activate or deactivate only directly granted roles.
  For example, in the hierarchy shown in the Mutual exclusivity at membership figure, assume that you have been granted the “specialist” role. You have all the permissions of the “specialist” role, and, implicitly, because of the hierarchy, you have all the permissions of the “doctor” and “consultant” roles. However, you can activate only the “specialist” role. You cannot activate “doctor” or “consultant” because they were not directly granted to you.
  Revoking roles from other roles is similar to granting roles to other roles. It removes a containment relationship, and the containment relationship must be a direct one.

For example:

- If the system security officer revokes the “doctor” role from “specialist,” “specialist” no longer contains the “consultant” role or the “intern” role.
- The system security officer cannot revoke the “intern” role from “specialist” because “intern” is not directly contained by “specialist.”

Related Information

Activate or Deactivate Roles [page 104]

5.7 Default Activation at Login

A system security officer can change role activation using `alter login` or `alter login profile`.

When a user logs in to SAP ASE, the user’s roles are not necessarily active, depending upon how the role is set up as a default role. If a role has a password associated with it, the user must use the `set role` command to activate the role.

The system security officer determines whether to activate roles granted by default at login and uses the `auto activated roles` attribute of `alter login profile` or `alter login` to set the default status of user roles individually for each user. Individual users can change only their own default settings. `auto activated roles` only affects user roles, not system roles.

By default, user-defined roles that are granted are not activated at login, but system roles that are granted are automatically activated, if they do not have passwords associated with them.

The following example shows how to automatically activate roles on login if they are not password protected.

```
alter login mgr add auto activated roles
```

```
mgr_role, eng_role
```

The following example shows how to use the login profile to automatically activate roles on login if they are non password protected. The `mgr_role` and `eng_role` must be granted to `mgr_lp`:

```
alter login profile mgr_lp add auto activated roles mgr_role, eng_role
```

5.8 Conditions for Role Activation

By specifying a `where` clause on the `grant role` statement, the system security officer can control the conditions under which a user can assume a given role.

SAP ASE evaluates the condition, termed a role activation predicate, when the user sets the role on or, for default or automatically activated roles, during login. Only if the role activation predicate evaluates to true is the role activated.

Related Information

Predicated Role Activation [page 231]

5.9 Drop User-Defined Roles

As system security officer, drop a role using the `drop role` command.

The syntax is:

```
drop role <role_name> [with override]
```

where `<role_name>` is the name of a user-defined role.

`with override` revokes all access privileges granted to the role in every database on the server.

If you do not use with the `override` option, you must revoke all privileges granted to the role in all databases before you can drop the role. If you do not, the command fails. To revoke privileges, use the `revoke` command

You need not drop memberships before dropping a role. Dropping a role automatically removes any user's membership in that role, regardless of whether you use the `with override` option.

5.10 Activate or Deactivate Roles

Roles must be active to have access privileges.

A default role is activated during login. Roles with passwords are always inactive at login. If the activation predicate on a default role evaluates to false during login, the role is silently ignored and remains inactive.

To activate or deactivate a role:

```
set role <role_name> [with passwd "<password>"] {on | off}
```

Include the `with passwd` parameter only if you are activating role. See the *Reference Manual: Commands*.

For example, to activate the "financial_analyst" role with the password "sailing19", enter:

```
set role financial_analyst with passwd "sailing19" on
```

If the role was granted using an activation predicate, the predicate is evaluated. If the predicate evaluates to true, the role is enabled; otherwise, the role remains inactive and the server returns an error message.

Activate roles only when you need them, and deactivate them when the roles are no longer necessary. Keep in mind that, when the `sa_role` is active, you assume the identity of database owner within any database that you use.

5.11 Display Information About Roles

Use these system procedures and functions to find information about roles.

To display information about	Use
The role ID of a role name	`role_id` system function
The role name of a role ID	`role_name` system function
System roles	`show_role` system function
Role hierarchies and roles that have been granted to a user or users	`sp_displayroles` system procedure
Whether one role contains another role in a role hierarchy	`role_contain` system function
Whether two roles are mutually exclusive	`mut_excl_roles` system function
Roles that are active for the current session	`sp_activeroles` system procedure
Whether you have activated the correct role to execute a procedure	`has_role` system function
Logins, including roles that have been granted	`sp_displaylogin` system procedure
Permissions for a user, group, or role	`sp_helprotect` system procedure

5.11.1 Find Role IDs and Names

To find a role ID or role name, use the `role_id` and `role_name` functions.

To find a role ID when you know the role name, use: `role_ID`:

```
role_id(<role_name>)
```

Any user can execute `role_id`. If the role is valid, `role_id` returns the server-wide ID of the role (`srid`). The `syssrvroles` system table contains an `srid` column with the role ID and a `name` column with the role name. If the role is invalid, `role_id` returns NULL.

To find a role name when you know the role ID, use: `role_name`:

```
role_name(<role_id>)
```

Any user can execute `role_name`.

5.11.2 View Active System Roles

Use `show_role` to display the currently active system roles for the specified login.

The syntax is:

```
show_role()
```

If you have not activated any system role, `show_role` returns NULL. If you are a database owner, and you execute `show_role` after using `setuser` to impersonate another user, `show_role` returns your own active system roles, not those for whom you are impersonating.

Any user can execute `show_role`.

> **i** Note
>
> The `show_role` function does not include information about user-defined roles.

5.11.3 Display Role Hierarchy

You can use `sp_displayroles` to see all roles granted to your login name or see the entire hierarchy tree of roles displayed in table format.

The syntax is:

```
sp_displayroles {login_name | <rolename >[, expand_up | expand_down]}
```

Any user can execute `sp_displayroles` to see his or her own roles. Only the system security officer can view information about roles granted to other users.

5.11.4 View User Roles in a Hierarchy

Use `role_contain` to determine whether any role you specify contains any other role you specify.

The syntax is:

```
role_contain (["<role1>", "<role2>"])
```

If `<role1>` is contained by `<role2>`, `role_contain` returns 1.

Any user can execute `role_contain`.

5.11.5 Determine Mutual Exclusivity

Use the `mut_excl_roles` function to determine whether any two roles assigned to you are mutually exclusive, and the level at which they are mutually exclusive.

The syntax is:

```
mut_excl_roles(<role1>, <role2>, {membership | activation})
```

Any user can execute `mut_excl_roles`. If the specified roles, or any role contained by either specified role, are mutually exclusive, `mut_excl_roles` returns 1; if the roles are not mutually exclusive, `mut_excl_roles` returns 0.

5.11.6 Determine Role Activation

To find all active roles for the current login session, use `sp_activeroles`.

The syntax is:

```
sp_activeroles [expand_down]
```

`expand_down` displays the hierarchy of all roles contained by any roles granted to you.

Any user can execute `sp_activeroles`.

5.11.7 Check for Roles in Stored Procedures

Use `has_role` within a stored procedure to allow only users with a specific role to execute the procedure.

`has_role` prevents inappropriate access to a particular stored procedure.

You can use `grant execute` to grant execute permission on a stored procedure to all users who have been granted a specified role. Similarly, `revoke execute` removes this permission.

However, `grant execute` permission does not prevent users who do not have the specified role from being granted execute permission on a stored procedure. To ensure, for example, that all users who are not system administrators cannot execute a stored procedure, use `has_role` within the stored procedure to check whether the invoking user has the correct role to execute the procedure.

`has_role` takes a string for the required role and returns 1or 2 if the invoker possesses it. Otherwise, it returns 0.

For example, here is a procedure that uses `has_role` to determine if the user has `sa_role` role:

```
create proc test_proc
as
if (has_role("sa_role",0) = 0)
begin
    print "You do not have the correct role"
    return -1
end
else
    print "You have System Administrator role"
    return 0
```

5.12 Grant or Revoke Roles

After a role is defined, it can be granted to any login account or role on the server, provided that it does not violate the rules of mutual exclusivity and hierarchy.

Task	Required role	Command
Grant the `sa_role` role	System administrator	`grant role`
Grant the `sso_role` role	System security officer	`grant role`
Grant the `oper_role` role	System security officer	`grant role`
Grant user-defined roles	System security officer	`grant role`
Create role hierarchies	System security officer	`grant role`
Modify role hierarchies	System security officer	`revoke role`
Revoke system roles	System security officer	`revoke role`
Revoke user-defined roles	System security officer	`revoke role`

5.12.1 Grant Roles

To grant roles to users, roles, or login profiles, use the `grant role` command.

The syntax is:

```
grant role <role_name>
    [where <pred_expression>]    to {<username> | <rolename> |
<login_profile_name> }
```

All roles listed in the `grant` statement are granted to all grantees. If you grant one role to another, it creates a role hierarchy.

For example, to grant Susan, Mary, and John the "financial_analyst" and the "payroll_specialist" roles, enter:

```
grant role financial_analyst, payroll_specialist
  to susan, mary, john
```

> **i** Note
>
> A role granted to a login profile can be activated by any user assigned that profile.

See `grant role` in the *Reference Manual: Commands*.

Related Information

Grant Roles to Login Profiles [page 66]

5.12.2 Understanding the grant Command and Roles

Use the `grant` command to grant permission on objects to all users who have been granted a specified role, whether system or user-defined.

This allows you to restrict use of an object to users who have been granted any of these roles:

- Any system-defined role
- Any user-defined role

A role can be granted only to a login account, another role, or login profile.

Grant permission to a role does not prevent users who do *not* have the specified role from being granted the same permission, directly or through a group. To ensure, for example, that only system administrators can successfully execute a stored procedure, use the `has_role` system function within the stored procedure itself to check that the user has been granted and has activated the requisite role.

Permissions granted to roles override permissions granted to users or groups. For example, assume John has been granted the system security officer role, and `sso_role` has been granted permission on the `sales` table. If John's individual permission on `sales` is revoked, he can still access `sales` when he has `sso_role` active because his role permissions override his individual permissions.

Related Information

Display Information About Roles [page 104]

5.12.3 Revoke Roles

Use `revoke role` to revoke roles from users, other roles, and login profiles.

The syntax is:

```
revoke role <role_name> [{<,> <role_name>}...]from <grantee> [{<,> <grantee>}...]
```

All roles listed in the `revoke` statement are revoked from all grantees.

5.13 Secure Role Passwords

In versions of SAP ASE earlier than 15.7, role passwords were stored using proprietary encryption in the `syssrvroles` system table. In version 15.7, role passwords are stored securely on-disk as SHA-256 digests.

When you upgrade to version 15.7 or later, and activate a role password for the first time after the upgrade, A the role password is encrypted and stored as an SHA-256 digest.

You cannot downgrade a role password that has been encrypted in SHA-256; instead, upon downgrade, the role password is truncated and the role is locked. The administrator must then reset the password and unlock the role after the downgrade.

> **i** Note
>
> In a high availability environment, those role passwords that are upgraded on first use on a primary server are also upgraded on its companion server.

5.13.1 Character Set Considerations

Passwords are automatically converted to a canonical—that is, a universal standardized form.

This automatic conversion prevents role-activation failures due to mismatched character mapping when you change the default character set. In versions earlier than 15.7, passwords used the server's default character set before they were encrypted.

5.13.2 Locked Roles and syssrvroles

You can configure a role to lock automatically after a certain number of failed role-activation attempts using the `max failed_logins` option, or manually using `alter role <rolename> lock`.

Information about locked roles is stored in the `syssrvroles` system table:

- `lockdate` – indicates when the role was locked. `lockdate` is set to the `datetime` when the role was locked.

- `locksuid` – indicates who locked the role.
- `lockreason` – indicates why the role was locked. `lockreason` is coded into an integer that can be represented with an internationalized message. Each reason has a message in the `MSGDB` database added to identify the reason in the local language.

These are reset to NULL when a role is unlocked.

The values and descriptions are:

Values for `lockreason`	**Value for** `locksuid`	**Description of** `lockreason` **of role**
NULL	NULL	Role has not been locked
1	`suid` of caller of `alter role`	Role locked by `suid` by manually executing `alter role <rolename> lock`
2	`suid` of user whose last attempted role activation led to the role getting locked	Role locked due to failed-role activation attempts reaching maximum number of failed logins

> **i** Note
>
> If you are using high availability functionality, both the primary and companion servers are updated when you update the `syssrvroles` columns.

5.13.3 Login Password Policy Checks Applied to Role Passwords

Password complexity options that are applicable to login passwords are also applied to role passwords.

The following options check which are extended to role passwords:

- `disallow simple passwords`
- `min digits in password`
- `min alpha in password`
- `min special char in password`
- `min upper char in password`
- `min lower char in password`
- `systemwide password expiration`
- `password exp warn interval`
- `minimum password length`
- `maximum failed logins`
- `expire login`

5.13.3.1 High-Availability Support for Password Policy Options

The high-availability functionality synchronizes password policy options between primary and secondary servers.

These are the supported password options:

- `disallow simple passwords`
- `min digits in password`
- `min alpha in password`
- `min special char in password`
- `min upper char in password`
- `min lower char in password`
- `systemwide password expiration`
- `password exp warn interval`
- `minimum password length`
- `maximum failed login`
- `expire login`
- `keypair regeneration period`
- `keypair error retry wait`
- `keypair error retry count`

SAP ASE uses a "`password policy`" quorum attribute to check the inconsistency of values on both the primary and secondary servers. A high-availability advisory check succeeds when all those value are the same on both servers, and fail when the values differ. For example:

```
sp_companion "MONEY1", do_advisory, 'all'
go
```

```
Attribute Name          Attrib Type  Local Value  Remote Value  Advisory
--------------------  -------------  -----------  ------------  ------------
expire login            password po   1            0             2
maximum failed          password po   3            5             2
min alpha in password   password po   10           12            2
```

A value of 2 set in the `advisory` column of the output indicates that the user cannot proceed with the cluster operation unless the values on both the companions match.

The output of `sp_companion do_advisory` also indicates the inconsistency in any of the particular password policy checks on both servers.

5.13.4 Configure SAP ASE for Roles

Before using the role functionality, make sure SAP ASE has additional disk space in the `master` database and transaction log for the columns added to the `syssrvroles` table.

A database administrator can use the `alter database` command to add the additional space.

To identify the density of roles per page and log space you need for role password changes, use `sp_help syssrvroles` and `sp_helpdb`. You can then compare the values of:

- The values of the log space from before and after a specific number of password changes
- The specific number of `set role with passwd` commands that update `syssrvroles` with dates

5.13.4.1 Upgrade SAP ASE

During the upgrade process, SAP ASE automatically adds the columns `locksuid`, `lockreason`, and `lockdate` into `syssrvroles`.

These columns are nullable, and have a default value of NULL after the upgrade. Values are set only when needed.

5.13.4.2 Downgrade SAP ASE

When you downgrade SAP ASE, role passwords are truncated and locked.

In addition, SAP ASE does not support the use of `allow password downgrade` for role passwords. After a downgrade, the administrator should reset the role passwords and unlock the role accounts before using them again.

During the downgrade process, SAP ASE:

- Truncates role passwords and locks roles
- Removes any attributes in `sysattributes` under class 35, as well as class 35 itself
- Removes `locksuid`, `lockreason`, and `lockdate` columns from `syssrvroles`

The actions to downgrade a password occur when you execute `sp_downgrade_esd`. in single-user mode. A `dataserver` started with a "`-m`" command line option starts the server in single-user mode and allows only the system administrator to log in.

In this example, executing `sp_downgrade_esd` results in the password of the "doctor_role" role becoming locked and truncated. The administrator can redirect this output to a file so that the passwords for these roles can be reset:

```
1> sp_downgrade_esd 'downgrade','16.1',1
2> go
```

```
 Checking databases for downgrade readiness.
There are no errors which involve encrypted columns.
Executing downgrade step 2 [dbcc markprocs(@dbid)] for :
- Database: master (dbid: 1)
sql command is: dbcc markprocs(@dbid)
...
Executing downgrade step 26 [delete statistics syssrvroles(password) if exists
(select 1
from syssrvroles where password is not
null) begin print "Truncating password and locking following role(s)" select
name from
syssrvroles where password is not null update
```

```
syssrvroles set password = null, status = (status | @lockrole) where password is
not null
end update syscolumns set length = 30
where id = object_id('syssrvroles') and name = 'password' update syssrvroles set
locksuid
= null, lockreason = null, lockdate = null
where locksuid is not null or lockreason is not null or lockdate is not null
delete
syscolumns where id = object_id('syssrvroles')
and name in ('locksuid', 'lockreason', 'lockdate')] for :
- Database: master (dbid: 1)
sql command is: delete statistics syssrvroles(password) if exists (select 1 from
syssrvroles where password is not null) begin print
"Truncating password and locking following role(s)" select name from syssrvroles
where
password is not null update syssrvroles set
password = null, status = (status | @lockrole) where password is not null end
update
syscolumns set length = 30 where id =
object_id('syssrvroles') and name = 'password' update syssrvroles set locksuid =
null,
lockreason = null, lockdate = null where
locksuid is not null or lockreason is not null or lockdate is not null delete
syscolumns
where id = object_id('syssrvroles') and
name in ('locksuid', 'lockreason', 'lockdate')
Truncating password and locking following role(s)
 name
 ------------------------------------------------------------
 doctor_role
Executing downgrade step 27 [delete sysattributes where class = 35 delete
sysattributes
where class = 39 update syslogins set lpid =
null, crsuid = null where lpid is not null or crsuid is not null delete
syscolumns where
id = object_id('syslogins') and name in
('lpid', 'crsuid') delete syslogins where (status & @lp_status) = @lp_status
update
syslogins set status = status & ~(@exempt_lock)
where (status & @exempt_lock) = @exempt_lock] for :
- Database: master (dbid: 1)
sql command is: delete sysattributes where class = 35 delete sysattributes where
class =
39 update syslogins set lpid = null, crsuid
= null where lpid is not null or crsuid is not null delete syscolumns where id =
object_id('syslogins') and name in ('lpid',
'crsuid') delete syslogins where (status & @lp_status) = @lp_status update
syslogins set
status = status & ~(@exempt_lock) where
(status & @exempt_lock) = @exempt_lock
...
(return status = 0)
```

Additional messages appear in the error log to identify steps that occurred during `sp_downgrade_esd` and any system errors that may occur, such as in this example of error log output for the example downgrade procedure:

```
00:0006:00000:00006:2011/06/28 06:21:24.12 server  Starting downgrading ASE.
00:0006:00000:00006:2011/06/28 06:21:24.12 server  Downgrade : Marking stored
procedures
to be recreated from text.
00:0006:00000:00006:2011/06/28 06:21:26.13 server  Downgrade : Removing full
logging
modes from sysattributes.
00:0006:00000:00006:2011/06/28 06:21:26.13 server  Downgrade : Downgrading data-
only
```

```
locked table rows.
00:0006:00000:00006:2011/06/28 06:21:26.13 server  Downgrade : Removing full
logging
modes from sysattributes.
00:0006:00000:00006:2011/06/28 06:21:26.13 server  Downgrade : Removing column
sysoptions.number.
00:0006:00000:00006:2011/06/28 06:21:26.13 server  Downgrade : Removing
srvprincipal
column from sysservers system table
00:0006:00000:00006:2011/06/28 06:21:26.14 server  Downgrade : Removing
'automatic master
key access' configuration parameter.
00:0006:00000:00006:2011/06/28 06:21:26.14 server  Downgrade : Removing
DualControl
sysattribute rows
00:0006:00000:00006:2011/06/28 06:21:26.14 server  Downgrade : Downgrading
sysattributes
system table.
00:0006:00000:00006:2011/06/28 06:21:26.16 server  Downgrade : Downgrading
syscomments
system table.
00:0006:00000:00006:2011/06/28 06:21:26.19 server  Downgrade : Truncated role
password,
locked role and removed columns locksuid, lockreason, lockdate from syssrvroles
00:0006:00000:00006:2011/06/28 06:21:26.21 server  Downgrade : Removing catalog
changes
for RSA Keypair Regeneration Period and Login Profile
00:0006:00000:00006:2011/06/28 06:21:26.21 server  Downgrade : Turning on
database
downgrade indicator.
00:0006:00000:00006:2011/06/28 06:21:26.21 server  Downgrade : Resetting
database version
indicator.
00:0006:00000:00006:2011/06/28 06:21:26.21 server  ASE downgrade completed.
```

After running `sp_downgrade_esd`, shut down the server to avoid new logins or other actions that may modify data or system catalogs.

You can enable locked roles and truncated password. In this example, the output of `sp_displayroles` shows that the downgrade process has locked "doctor_role" and truncated its password:

```
select srid,status,name,password from syssrvroles
go
```

```
suid   status  name                password
------ ------- ------------------- -----
33     2       doctor_role         NULL
```

This unlocks the role:

```
alter role doctor_role unlock
```

This sets a new password for the role:

```
alter role doctor_role add passwd "dProle1"
```

Running `sp_displayroles` now displays that the role is unlocked and has a password:

```
select srid,status,name,
"vers"=substring(password,2,1) from syssrvroles
go
```

```
suid   status  name                vers
------ ------- ------------------- -----
33     0       doctor_role         0x05
```

6 External Authentication

You can authenticate users with authentication data stored in repositories that are external to SAP ASE.

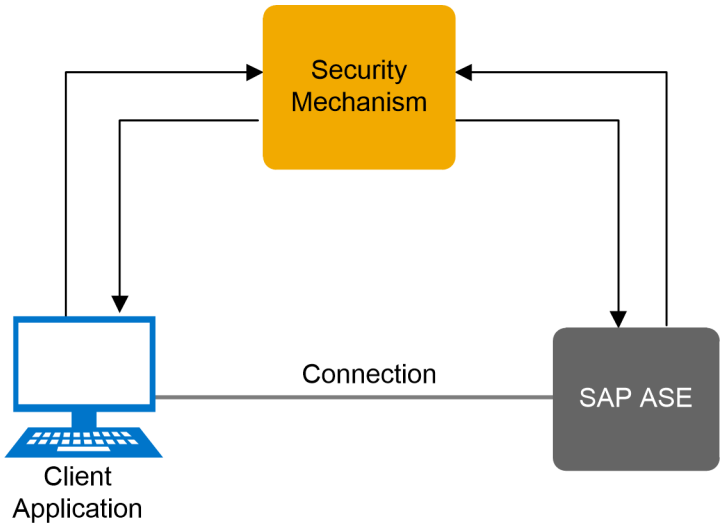
You can enhance the security for large, heterogeneous applications by authenticating logins with a central repository. These external authentication methods are supported:

- Kerberos – provides a centralized and secure authentication mechanism in enterprise environments that employ the Kerberos infrastructure. Authentication occurs with a trusted, third-party server called a key distribution center (KDC) that verifies both the client and the server.
- LDAP user authentication – Lightweight Directory Access Protocol (LDAP) provides a centralized authentication mechanism based on a user's login name and password.
- PAM user authentication – Pluggable Authentication Module (PAM) provides a centralized authentication mechanism that uses interfaces provided by the operating system for administration and runtime application interfaces.

> **i** Note
>
> To use these external authentication methods, you must obtain the ASE_SECDIR security and directory services license package.

6.1 Configure SAP ASE for Network-Based Security

The secure connection between a client and a server can be used for login authentication and message protection.



If a client requests authentication services:

1. The client validates the login with the security mechanism. The security mechanism returns a credential, which contains security-relevant information.

2. The client sends the credential to SAP ASE.
3. SAP ASE authenticates the client's credential with the security mechanism. If the credential is valid, a secure connection is established between the client and SAP ASE.

If the client requests message protection services:

1. The client uses the security mechanism to prepare the data packet it sends to SAP ASE.
   Depending upon which security services are requested, the security mechanism might encrypt the data or create a cryptographic signature associated with the data.
2. The client sends the data packet to SAP ASE.
3. Upon receiving the data packet, SAP ASE uses the security mechanism to perform any required decryption and validation.
4. SAP ASE returns results to the client, using the security mechanism to perform the security functions that were requested; for example, SAP ASE may return the results in encrypted form.

6.1.1 SAP ASE Security Services

Depending on the security mechanism you choose, SAP ASE allows you to use one or more security services.

You can use:

- Unified login – authenticates users once, without requiring them to supply a name and password every time they log in to an SAP ASE server.
- Message confidentiality – encrypts data over the network.
- Mutual authentication – verifies the identity of the client and the server. Mutual authentication can be requested only by the client; it cannot be required.
- Message integrity – verifies that data communications have not been modified.
- Replay detection – verifies that data has not been intercepted by an intruder.
- Out-of-sequence check – verifies the order of data communications.
- Message origin checks – verifies the origin of the message.
- Credential delegation – allows the client to delegate the credential to the server to enable secure connection with remote servers. This service is supported by Kerberos security mechanism. This currently supported for connections to remote servers through CIS.
- Remote procedure security – establishes mutual authentication, message confidentiality, and message integrity for remote procedure communications through CIS for Kerberos connections.

> **i Note**
>
> The security mechanism you are using may not employ all of these services.

Related Information

Obtain Information About Available Security Services [page 131]

6.1.2 Administering Network-Based Security

How to administer network-based security.

Procedure

1. Edit the configuration files: `libtcl.cfg`, `objectid.dat`, and `interfaces` (or directory service).

   See *Configuration Files for Security* and the *Open Client/Server Configuration Guide* for your platform.
2. Make sure the security administrator for the security mechanism has created logins for each user and for the SAP ASE and Backup Server.

   See *Identifying Users and Servers to the Security Mechanism* and the documentation supplied with your security mechanism.
3. Use `sp_configure` to configure security for your installation.
4. Restart the server to activate the `use security services` parameter.
5. Use `create login` to add login accounts to support enterprise-wide login. Optionally, specify a default secure login with `sp_configure`.

   See *Adding Logins to Support Unified Login*.
6. Use `security mechanism` option of `sp_serveroption` to enable security mechanism for required remote servers.

   See *Establish Kerberos Security for Remote Connections*.
7. Use `isql_r` or Open Client Client-Library to connect to SAP ASE, specifying the security services you want to use.

   See *Connect to the Server and Using the Security Services*. Also see *Security Features* in the *Open Client Client-Library/C Reference Manual* and the *Open Client/Server Configuration Guide* for your platform.
8. Use the functions `show_sec_services` and `is_sec_services_on` to check which security services are available. For a list of security mechanisms and their security services supported by SAP ASE, use `select` to query the `syssecmechs` system table.

6.1.3 Configuration Files for Security

Configuration files are created during installation at a default location in the directory structure.

These are the names and locations for configuration files:

File name	Description	Location
`libtcl.cfg` (`libtcl64.cfg` for 64-bit)	The driver configuration file contains information regarding directory, security, and network drivers, and any required initialization information.	*UNIX platforms*: `$SYBASE/$SYBASE_OCS/config` *Windows platforms*: `%SYBASE%\%SYBASE_OCS%\ini`

File name	Description	Location
`objectid.dat`	The object identifiers file maps global object identifiers to local names for character set, collating sequence, and security mechanisms.	*UNIX platforms*: `$SYBASE/config` *Windows platforms*: `%SYBASE%\ini`
UNIX: `interfaces` *Desktop platforms*: `sql.ini`	The `interfaces` file contains connection and security information for each server listed in the file.	*UNIX platforms*: `$SYBASE` *Desktop platforms*: `SYBASE_home\ini`

For a detailed description of the configuration files, see the *Open Client/Server Configuration Guide* for your platform.

6.1.3.1 Security Information for the Server

Use an `interfaces` file or a Directory Service to provide information about the servers in your installation.

The `interfaces` file contains network and security information for servers. To use security services, the `interfaces` file must include line for "secmech" that specifies the global identifier or identifiers of the security services you plan to use.

SAP ASE supports Directory Services to keep track of information about servers. A Directory Service manages the creation, modification, and retrieval of information about network servers. The advantage of using a Directory Service is that you do not need to update multiple `interfaces` files when a new server is added to your network or when a server moves to a new address. To use security services with a Directory Service, you must define the `secmech` security attribute to point to one or more global identifiers of the security services you plan to use.

UNIX Tools for Specifying the Security Mechanism

To specify the security mechanism or mechanisms:

- If you are using the `interfaces` file, use the `dscp` utility.
- If you are using a Directory Service, use the `dscp_r` utility.

> **i** Note
>
> The `dsedit` tool, which helps you create entries for either the `interfaces` file or a Directory Service, is available on UNIX platforms. However, it does not support the creation of `secmech` entries for security mechanisms.

For more information about `dscp`, see the *Open Client/Server Configuration Guide for UNIX*.

Desktop Tools for Specifying Server Attributes

To provide information about the servers for your installation in the `sql.ini` file or a Directory Service, use the `dsedit` utility. This utility provides a graphical user interface for specifying server attributes such as the server version, name, and security mechanism. For the security mechanism attribute, you can specify one or more object identifiers for the security mechanisms you plan to use. For information about using `dsedit`, see the *Open Client/Server Configuration Guide for Desktop Platforms*.

6.1.3.2 Prepare libtcl.cfg to Use Network-Based Security

Edit the `libtcl.cfg` or `libtcl64.cfg` for network-based security.

The `libtcl.cfg` and `libtcl64.cfg` (for 64-bit applications) contain information about three types of drivers:

- Network (Net-Library)
- Directory Services
- Security

A driver is a library that provides an interface to an external service provider. Drivers are dynamically loaded so that you can change the driver used by an application without relinking the application.

Entries for Network Drivers

The syntax for a network driver entry is:

```
<driver>=<protocol> <description>
```

where:

- `<driver>` – is the name of the network driver.
- `<protocol>` – is the name of the network protocol.
- `<description>` – is a description of the entry. This element is optional.

> **i** Note
>
> If you do not specify a network driver, an appropriate driver for your application and platform is automatically used. For example, for UNIX platforms, a driver that can handle threads is automatically chosen when security services are being used.

Entries for Directory Services

Directory Services entries apply if you want to use a Directory Service instead of the `interfaces` file. See the configuration documentation for your platform, and the *Open Client/Server Configuration Guide* for your platform.

Entries for Security Drivers

The syntax for a security driver entry is:

```
<provider>=<driver> <init-string>
```

where:

- `<provider>` – is the local name for the security mechanism. The mapping of the local name to a global object identifier is defined in `objectid.dat`.
  The default local names are:
  - "csfkrb5" – for the CyberSAFE or MIT Kerberos security mechanism.
  - "LIBSMSSP" – for Windows LAN Manager on Windows NT or Windows 95 (clients only).

  If you use a local mechanism name other than the default, change the local name in the `objectid.dat` file.
- `<driver – >`is the name of the security driver. The default location of all drivers for UNIX platforms is `$SYBASE/$SYBASE_OCS/lib`. The default location for Windows platform is `%SYBASE%\%SYBASE_OCS%\dll`.
- `<init-string>` – is an initialization string for the driver. This element is optional. The value for `<init-string>` varies by driver:
  - Kerberos driver – the following is the syntax for `<init-string>`, where `<realm>` is the default Kerberos realm name:

    ```
    secbase=@<realm>
    ```
  - Windows NT LAN Manager – `<init-string>` is not applicable.

UNIX Platform Information

No special tools for editing the `libtcl.cfg` file are available. Use your favorite editor to comment and uncomment the entries that are already in place after you install SAP ASE.

After you install SAP ASE on a UNIX platform, the `libtcl.cfg` file already contains entries for the three sections of the file:

- [DRIVERS]
- [DIRECTORY]
- [SECURITY]

The sections do not have to be in a specific order.

Make sure that the entries you do not want to use are commented (begin with ";") and the entries you want are uncommented (do not begin with ";").

For more information, see the *Open Client/Server Configuration Guide for UNIX*

Sample libtcl.cfg for Sun Solaris

```
[DRIVERS]
;libtli.so=tcp unused ; This is the non-threaded tli driver.
;libtli_r.so=tcp unused ; This is the threaded tli driver.
[SECURITY]
csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/krb5/lib/libgss.so
```

This file does not use Directory Services because all [DIRECTORY] section entries are commented.

Because all entries in the [DRIVERS] section for network drivers are also commented, appropriate drivers are automatically chosen by the system. SAP ASE automatically chooses a threaded driver when you use security services, and chooses an unthreaded driver for applications that cannot work with threaded drivers. For example, Backup Server does not support security services and does not work with a threaded driver.

Desktop Platform Information

The `ocscfg` utility automatically creates section headings for the `libtcl.cfg` file; you can also use `osccfg` to edit the `libtcl.cfg` file.

This is a sample `libtcl.cfg` file for desktop platforms:

```
[NT_DIRECTORY]
ntreg_dsa=LIBDREG  ditbase=software\sybase\serverdsa
[DRIVERS]
NLWNSCK=TCP  Winsock TCP/IP Net-Lib driver
NLMSNMP=NAMEPIPE  Named Pipe Net-Lib driver
NLNWLINK=SPX  NT NWLINK SPX/IPX Net-Lib driver
NLDECNET=DECNET  DecNET Net-Lib driver
[SECURITY]
NTLM=LIBSMSSP
```

See the *Open Client/Server Configuration Guide for Desktop Platforms*.

6.1.3.3 objectid.dat File

The `objectid.dat` file maps global object identifiers to local names and the file contains sections such as [CHARSET] for character sets and [SECURITY] for security services.

This is a sample `objectid.dat` file:

```
secmech]
       1.3.6.1.4.1.897.4.6.3   = NTLM
```

```
        1.3.6.1.4.1.897.4.6.6    = csfkrb5
```

The mapping of global object identifiers to local names, for example, is the Kerberos service (such as the identifier 1.3.6.1.4.1.897.4.6.6) to local names, such as "csfkrb5".

Use a text editor to change this file only if you have changed the local name of a security service in the `libtcl.cfg` file.

For example, if you changed:

```
[SECURITY]
csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/krb5/lib/libgss.so
```

to:

```
[SECURITY]
csfkrb5_group=libsybskrb.so secbase=@MYREALM libgss=/krb5/lib/libgss.so
```

Change the `objectid.dat` in `libtcl.cfg` to reflect the change. Simply change the local name in the line for Kerberos in `objectid.dat`:

```
1.3.6.1.4.1.897.4.6.6 = csfkrb5_group
```

> **i** Note
>
> You can specify only one local name per security mechanism.

6.1.4 Identify Users and Servers to the Security Mechanism

The security administrator for the security mechanism must define principals (both users and servers) to the security mechanism.

The tools you can use to add users and servers are:

- Kerberos – see your Kerberos vendor-specific tools for information about defining users and servers.
- Windows NT LAN Manager – run the User Manager tool to define users to the Windows NT LAN Manager. Define the SAP ASE server name as a user to Windows NT LAN Manager and display SAP ASE as that user name.

> **i** Note
>
> In a production environment, control access to files that contain the keys of the servers and users. If users can access the keys, they can create a server that impersonates your server.

See the documentation available from the third-party provider of the security mechanism for detailed information about how to perform required administrative tasks.

6.1.5 Configure SAP ASE for Security

Adaptive Server includes several configuration parameters for administering network-based security. To set these parameters, you must be a system security officer. All parameters for network-based security are part of the "Security-Related" configuration parameter group.

6.1.5.1 Enable Network-Based Security

To enable or disable network-based security, use `sp_configure` to set the `use security services` configuration parameter. .

If `use security services` is set to 1, SAP ASE supports a security mechanism when both of the following circumstances are true:

- The security mechanism's global identifier is listed in the `interfaces` file or Directory Service.
- The global identifier is mapped in `objectid.dat` to a local name that is listed in `libtcl.cfg`.

6.1.5.2 Require Users to be Authenticated

To require all users, other than the system security officer, to be authenticated by a security mechanism, set the `unified login required` configuration parameter to 1.

Only the user with the sso_role can log in to the server with a user name and password when this configuration parameter is set:

```
sp_configure "unified login required", [0|1]
```

For example, to require all logins to be authenticated by a security mechanism, execute:

```
sp_configure "unified login required", 1
```

6.1.5.3 Establish a Secure Default Login

SAP ASE uses a default login for any user who is not defined in `master..syslogins`, but who is preauthenticated by a security mechanism.

When a user with a valid credential from a security mechanism logs in to SAP ASE, the server checks whether the user name exists in `master..syslogins`. If it does, SAP ASE uses that user name. For example, if a user logs in to the Kerberos security mechanism as "ralph," and "ralph" is in `master..syslogins`, SAP ASE uses all roles and authorizations defined for "ralph" in the server.

However, if a user with a valid credential logs in to SAP ASE, but is unknown to the server, the login is accepted only if a secure default login is defined with `sp_configure`. SAP ASE uses the default login for any user who is not defined in `master..syslogins`, but who is preauthenticated by a security mechanism. The syntax is:

```
sp_configure "secure default login", 0, <login_name>
```

The default value for `secure default login< >`is "guest."

A secure default login must also be a valid login in `master..syslogins`. For example, to set the "gen_auth" as the default login:

1. Use `create login` to add the login as a valid user:

   ```
   create login gen_auth with password pwgenau
   ```

   This procedure sets the initial password to "pwgenau".
2. Designate the login as the security default:

   ```
   sp_configure "secure default login", 0, gen_auth
   ```

   SAP ASE uses this login for a user who is preauthenticated by a security mechanism but is unknown to SAP ASE.

   > **i** Note
   >
   > More than one user can assume the `suid` associated with the secure default login. Therefore, you might want to activate auditing for all activities of the default login. You may also want to consider using `create login` to add all users to the server.

6.1.5.4 Map Security Mechanism Login Names to Server Names

All login names must be valid identifiers.

Some security mechanisms may allow login names that are invalid in Adaptive Server. For example, login names that are longer than 30 characters, or login names containing special characters such as !, %, *, and & are invalid in Adaptive Server.

This table shows how invalid characters in login names are converted :

Invalid characters	Converts to
• Ampersand & • Apostrophe ' • Backslash \ • Colon : • Comma , • Equals sign = • Left quote ‘ • Percent % • Right angle bracket > • Right quote ’ • Tilde ~	Underscore _
• Asterisk * • Minus sign - • Pipe | • Plus sign + • Quotation marks " • Semicolon ; • Slash / • Square brackets [ ]	Pound sign #

See *Expressions, Identifiers, and Wildcard Characters*, in the *Reference Manual*.

6.1.5.5 Require Message Confidentiality with Encryption

To require all messages into and out of SAP ASE to be encrypted, set the `msg confidentiality reqd` configuration parameter to 1.

If this parameter is 0 (the default), message confidentiality is not required but may be established by the client. The syntax is:

```
sp_configure <configuration_parameter>, [0 | 1]
```

For example, to require that all messages be encrypted, execute:

```
sp_configure "msg confidentiality reqd", 1
```

6.1.5.6 Require Data Integrity

SAP ASE allows you to use the `msg integrity reqd` configuration parameter to require that one or more types of data integrity be checked for all messages.

Set `msg integrity reqd` to 1 to require that all messages be checked for general tampering. If `msg integrity reqd` is 0 (the default), message integrity is not required but may be established by the client if it is supported by the security mechanism.

6.1.5.7 Memory Requirements for Network-Based Security

Allocate approximately 2K additional memory per secure connection. The value of the `max total_memory` configuration parameter specifies the amount of memory that the server requires at start-up.

For example, if your server uses 2K logical pages, and if you expect the maximum number of secure connections occurring at the same time to be 150, increase the `max total_memory` parameter by 150, which increases memory allocation by 150 2K blocks.

The syntax is:

```
sp_configure "max total_memory", <value>
```

For example, if the server requires 75,000 2K blocks of memory, including the increased memory for network-based security, execute:

```
sp_configure "max total_memory", 75000
```

See *Configuring Memory*,in *System Administration Guide: Volume 2*.

6.1.5.8 Using NT Lan Manager Security Services on Windows 64-bit

NT Lan Manager security services are supported on Windows 64-bit.

Context

In order to use the NTLM Security Services on Windows 64bit, changes are required in the `libtcl64.cfg`, `libtcl.cfg`, and `sql.ini` files.

Procedure

1. Update the `[SECURITY]` section of the `%SYBASE%\%SYBASE_OCS%\ini\libtcl64.cfg` file.
   a. Add the following to the `[SECURITY]` section:

   ```
   NTLM=LIBSYBSMSSP64
   ```

   This change allows 64-bit ASE and 64-bit Open Client applications to use the security driver library `libsybsmssp64.dll` at runtime. The `libsybsmssp64.dll` library is located in `%SYBASE%\%SYBASE_OCS%\dll`, along with other Open Client dynamic link libraries.
2. Update the `[SECURITY]` section in the `%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg` file.
   a. Add the following to the `[SECURITY]` section:

   ```
   NTLM=LIBSYBSMSSP
   ```

   The `libtcl.cfg` is used by the 32bit `isql` utility and 32-bit OpenClient applications.
3. Choose one of the following methods to specify the OID value for NTLM.
   - Update the `[SECMECH]` section of the `%SYBASE%\ini\sql.ini` file. Add the following to the `sql.ini` file:

     ```
     [ASENAME]
     master=TCP,<host>,<port>
     query=TCP,<host>,<port>
     secmech=1.3.6.1.4.1.897.4.6.3
     ```

   - Use the `dsedit` utility to add the 'Server Security' attribute value of `'1.3.6.1.4.1.897.4.6.3'` to your server.

     > **i** Note
     >
     > This OID value derived from the `%SYBASE%\ini\objectid.dat` file which should not be modified

     .
4. Make sure Adaptive Server is configured for security services. For example, to enable services with LAN Manager, execute:

   ```
   sp_configure "use security services", 1
   ```

   For more information, see *Using Security Services with NT LAN Manager* in the *Configuration Guide for Windows NT*.
5. Make sure you have a login on the Adaptive Server that corresponds with your Windows login.
6. Connect to the Adaptive Server without a user name and password. For example:
   - `isql -V -SASENAME`
   - `isql64 -V -SASENAME`

6.1.6 Add Logins to Support Unified Login

Consider whether you want to allow only those users who are defined as valid logins to use SAP ASE, or whether you want users to be able to log in with the default login.

When users log in with a preauthenticated credential, SAP ASE:

1. Checks whether the user is a valid user in `master..syslogins`. If the user is listed in `master..syslogins`, the login is accepted without requiring a password.
2. If the user name is not in `master..syslogins`, a check is performed as to whether a default secure login is defined. If the default login is defined, the user is logged in successfully using the default. If a default login is not defined, the user cannot log in.

To define a default login, add the default login in `master..syslogins` and use `sp_configure`.

6.1.6.1 General Procedure for Adding Logins

Add logins to the server and, optionally, to add users with appropriate roles and authorizations to one or more databases.

Context

Follow this general procedure to add logins to the server and, optionally, to add users with appropriate roles and authorizations to one or more databases.

Procedure

1. System security officer uses `create login` to add a login for the user.
   See *Create Login Accounts*.
2. The system administrator or database owner uses `sp_adduser` to add the user to one or more databases.
   See *Add Users to Databases*.
3. The system administrator or database owner uses `sp_changegroup` to add the user to group in the database.
   See *Change Group Membership*.
4. The system administrator or system security officer uses `grant role` to grant system roles to the user.
   See *Grant or Revoke Roles*.
5. The system security officer uses `create role` and `grant role` to create user-defined roles and grant the roles to users.
   See *Grant Roles*.
6. The Database object owners grant access to database objects.
   See *Manage User Permissions*.

Related Information

Create Login Accounts [page 26]
Add Users to Databases [page 72]
Change Group Membership [page 76]
Grant or Revoke Roles [page 107]
Grant Roles [page 107]
Manage User Permissions [page 167]

6.1.7 Connect to a Server Using Security Services

The `isql` and `bcp` utilities enable network-based security services on the connection.

The `isql` and `bcp` utilities include the following command line options:

- `-R <remote_server_principal>`
- `-V <security_options>`
- `-Z <security_mechanism>`

These options are described in the following paragraphs.

- `-R <remote_server_principal>` – specifies the principal name for the server as defined to the security mechanism. By default, a server's principal name matches the server's network name (which is specified with the `-S` option or the DSQUERY environment variable). The `-R` option must be used when the server's principal name and network name are not the same.
- `-V <security_options>` – specifies network-based user authentication. With this option, the user must log in to the network's security system before running the utility. In this case, if a user specifies the `-U` option, the user must supply the network user name known to the security mechanism; any password supplied with the `-P` option is ignored. `-V` can be followed by a `<security_options>` string of key-letter options to enable additional security services. These key letters are:
  - `c` – enables data confidentiality service.
  - `d` – requests credential delegation and forwards client credentials.
  - `i` – enables data integrity service.
  - `m` – enables mutual authentication for connection establishment.
  - `o` – enables data origin stamping service.
  - `r` – enables data replay detection.
  - `q` – enables out-of-sequence detection.
- `-Z <security_mechanism>` – specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the `libtcl.cfg` configuration file. If no `<security_mechanism>` name is supplied, the default mechanism is used. See the *Open Client/Server Configuration Guide* for your platform.

If you are using Client-Library to connect to SAP ASE, you can define security properties before connecting to the server. For example, to check message sequencing, set the CS_SEC_DETECTSEQ property. For information about using security services with Client-Library, see the *Open Client Client-Library/C Reference Manual*.

6.1.7.1 Security Mechanisms for the Client

SAP ASE, when it is started, determines the set of security mechanisms it supports. From the list of supported security mechanisms, SAP ASE must choose the one to be used for a particular client.

If the client specifies a security mechanism (for example with the `-Z` option of `isql`), SAP ASE uses that security mechanism. Otherwise, it uses the first security mechanism listed in the `libtcl.cfg` file.

6.1.8 Obtain Information About Available Security Services

You can determine which security mechanisms and services are supported, what security services are active for the current session, and whether a particular security service is enabled for the session

Determining Supported Security Services and Mechanisms

A system table, `syssecmechs`, provides information about the security mechanisms and security services supported. The table, which is dynamically built when you query it, contains these columns:

- `sec_mech_name` – is the name of the security mechanism; for example, the security mechanism might be "NT LANMANAGER."
- `available_service` – is the name of a security service supported by the security mechanism; for example, the security service might be "unified login."

The table may have several rows for a single security mechanism: one row for each security service supported by the mechanism.

To list all the security mechanisms and services supported, run:

```
select * from syssecmechs
```

Determining Active Security Services

To determine which security services are active for the current session, use the function `show_sec_services`:

```
select show_sec_services()
```

```
-------------------------------------------------
        unifiedlogin mutualauth confidentiality
(1 row affected)
```

Determining Whether a Security Service is Enabled

To determine whether a particular security service, such as "mutualauth" is enabled, use the function `is_sec_service_on`, where `<security_service_nm>` is a security service that is available:

```
is_sec_service_on(<security_service_nm>)
```

Use the security server that is returned when you query `syssecmechs`.

For example, to determine whether "mutualauth" is enabled, execute:

```
select is_sec_service_on("mutualauth")
```

```
-----------
          1
(1 row affected)
```

A result of 1 indicates the security service is enabled for the session. A result of 0 indicates the service is not in use.

6.1.9 Kerberos

Kerberos is a network authentication protocol that uses secret-key cryptography so that a client can prove its identity to a server across a network connection.

User credentials are obtained when the user logs in to the operating system, or by executing an authentication program. Each application uses these credentials to perform authentication. Users only have to log in once, instead of having to log in to each application.

Kerberos assumes the key distribution center (KDC) is running and properly configured for your realm, and the client libraries are installed under or on each client host in your realm. For configuration information, consult the documentation and the reference pages that come with the Kerberos software.

SAP ASE supports Kerberos through:

- CyberSafe Kerberos libraries, version 2.1
- MIT Kerberos libraries, version 1.4.3 (version 4.0.1 for Windows 64-bit)
- Native libraries

> **i** Note
>
> To enable Kerberos security options, you must have ASE_SECDIR, the "Security and directory services" package.

6.1.9.1 Concurrent Kerberos Authentication

SAP ASE supports concurrent Kerberos authentication.

When there are concurrent logins using Kerberos authentication, SAP ASE establishes multiple Kerberos authentication sessions.

SAP ASE versions earlier than 15.0.3 used locking mechanisms during Kerberos authentication to protect internal data structures.

6.1.9.2 Start an SAP ASE Server Under Kerberos

To start an SAP ASE server under Kerberos, add the server name to the KDC and extract the service key to a key table file.

For example:

```
/krb5/bin/admin admin/ASE -k -t /krb5/v5srvtab -R" addrn my_ase; mod
my_ase attr nopwchg; ext -n my_ase eytabfile.krb5"
Connecting as: admin/ASE
Connected to csfA5v01 in realm ASE.
Principal added.
Principal modified.
Key extracted.
Disconnected.
```

> **i** Note
>
> The administrator can also be authenticated using a password on the command line. In this example, the `-k` option is used, which tells the administrator to search the `/krb5/v5srvtab` file (specified using the `-t` option) for the administrator and the SAP ASE server key, instead of prompting for a password, which is useful for writing shell scripts.

6.1.9.3 Configuring Kerberos

The configuration process is similar, regardless of which variety of Kerberos you use.

Procedure

1. Set up Kerberos third-party software and create a Kerberos administrative user. To do this, you must:
   a. Install Kerberos client software on machines where Open Client Server clients or SAP ASE will run. The following client packages have been verified to work with:
      - CyberSafe TrustBroker 2.1
      - MIT Kerberos version 1.4.3 (version 4.0.1 for Windows 64-bit)

b. Install the Kerberos KDC server on a separate, dedicated machine.

   KDCs from CyberSafe TrustBroker version 2.1, MIT Kerberos version.1.8.2, and Microsoft Windows Active Directory have been verified for use with SAP ASE.

c. Create an administrator account with administration privileges on the Kerberos server. This account is used for subsequent client actions such as creating principals from the client machines.

Execute the remainder of these steps on the Kerberos client machine.

2. Add Kerberos principal for SAP ASE `<ase120srv>` or `<ase120srv@MYREALM>`.
3. Extract the `keytab` file for principal `<ase120srv@MYREALM>` and store it as a file:

```
/krb5/v5srvtab
```

   The following UNIX examples use the command line tool `kadmin`, available with CyberSafe or MIT Kerberos (there are also GUI tools available to administer Kerberos and users):

```
CyberSafe Kadmin:
% kadmin aseadmin
Principal - aseadmin@MYREALM
Enter password:
Connected to csfA5v01 in realm ASE.
Command: add ase120srv
Enter password:
Re-enter password for verification:
Principal added.
Command: ext -n ase120srv
Service Key Table File Name (/krb5/v5srvtab):
Key extracted.
Command: quit
Disconnected.
```

   In a production environment, control the access to the `keytab` file. If a user can read the `keytab` file, he or she can create a server that impersonates your server.

   Use `chmod` and `chgrp` so that `/krb5/v5srvtab` is:

```
-rw-r----- 1 root sybase 45 Feb 27 15:42 /krb5/v5srvtab
```

   When using Active Directory as the KDC, log in to the Domain Controller to add users and SAP ASE principals. Use the Active Directory Users and Computers wizard to guide you through creating users and principals.

   Extracting the `keytab` file for use with SAP ASE requires an optional tool called `ktpass`, which is included in the Microsoft Support Tools package.

   With Active Directory, extracting the `keytab` with `ktpass` is a separate step from creating the principal. The `keytab` file on Windows for SAP ASE is located with the CyberSafe program files. For example, `c:\Program Files\CyberSafe\v5srvtab` is the expected location of the SAP ASE `keytab` file when CyberSafe software is installed on the C: drive.
4. Add a Kerberos principal for the user “sybuser1” as “sybuser1@MYREALM”.
5. Start the SAP ASE server and use `isql` to log in as “sa”. The following steps configure SAP ASE parameters to use Kerberos security services, and create the user login account. These are the same on both Windows or UNIX machines:

a. Change configuration parameter `use security services` to 1:

```
sp_configure 'use security services', 1
```

b. Add a new login for user, “sybuser1” and then add the user:

```
create login sybuser1 with password <password>
```

6. Shut down the SAP ASE server and modify administrative files and connectivity configuration files.
   - On UNIX platforms – the `interfaces` file is under `$SYBASE/` and has an entry that looks similar to:

     ```
     ase120srv
           master tli tcp myhost 2524
           query tli tcp myhost 2524
           secmech 1.3.6.1.4.1.897.4.6.6
     ```

     On Windows platforms – the `sql.ini` file is in `%SYBASE%\ini`, and has an equivalent server entry that looks like:

     ```
     [ase120srv] master=TCP,myhost,2524 query=TCP,myhost,2524
     secmech=1.3.6.1.4.1.897.4.6.6
     ```

   - The `libtcl.cfg` or `libtcl64.cfg` file is located in `$SYBASE/$SYBASE_OCS/config/` on UNIX platforms. The SECURITY section should have an entry that looks similar to the following for CyberSafe Kerberos client libraries: (the lines starting with “csfkrb5” are single lines in these examples, but are split for space purposes)

     ```
     [SECURITY]
     csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/krb5/lib/libgss.so
     ```

     A 64-bit CyberSafe Kerberos client library entry follows :

     ```
     [SECURITY]
     csfkrb5=libsybskrb64.so secbase=@MYREALM libgss=/krb5/appsec-rt \
     /lib/64/libgss.so
     ```

     For a machine that uses MIT Kerberos client libraries, the entry looks something like:

     ```
     [SECURITY]
     csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/opt/mitkrb5/lib/ \
     libgssapi_krb5.so
     ```

     For a machine that uses Native OS provided libraries, such as Linux, it looks similar to:

     ```
     [SECURITY]
     csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/usr/kerberos/lib/ \
     libgssapi_krb5.so
     ```

     On Windows – the `%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg` file contains an entry like:

     ```
     [SECURITY]
     csfkrb5=libskrb secbase=@MYREALM libgss=C:\WinNT\System32\ ^
     gssapi32.dll
     ```

> **Note**
>
> The `libgss=<gss shared object path>` specifies the GSS API library to be used. You must distinctly locate the Kerberos Client libraries being used, especially when multiple versions are installed on a machine.

- Also check the `objectid.dat` under `$SYBASE/$SYBASE_OCS/config/` and make sure the `[secmech]` section has an entry for `csfkrb5`:

```
[secmech]
1.3.6.1.4.1.897.4.6.6 = csfkrb5
```

7. You can use environment variables to override default locations of `keytab` files, Kerberos configuration, and realm configuration files. This is Kerberos-specific behavior and may not work consistently on all platforms.

   For example, use the CSFC5KTNAME environment variable on CyberSafe UNIX platforms to specify the `keytab` file:

   ```
   % setenv CSFC5KTNAME /krb5/v5srvtab
   ```

   For MIT Kerberos, the equivalent environment variable is KRB5_KTNAME.

   See the vendor documentation for information about these environment variables.

   You may need to modify the environment variable for dynamic library search paths. On UNIX, the most commonly used environment variable is LD_LIBRARY_PATH; on Windows, PATH is typically set to include DLL locations. You may need to modify these environment variables to enable applications to load the third-party objects correctly. For example, this command adds the location of CyberSafe 32-bit `libgss.so` shared object to the search path in a C-shell environment:

   ```
   % set path = ( /krb5/lib $path )
   ```

8. Restart the server. You should see:

   ```
   00:00000:00000:2001/07/25 11:43:09.91 server
   Successfully initialized the security mechanism
   'csfkrb5'. The SQL Server will support use of this
   security mechanism.
   ```

9. Use `isql_r` as UNIX user “sybuser1” (without the `-U` and `-P` arguments) to connect:

   ```
   % $SYBASE/$SYBASE_OCS/bin/isql_r -Sase120srv -V
   1>...
   ```

   You can also use the encryption option:

   ```
   $SYBASE/$SYBASE_OCS/bin/isql_r -Sase120srv -Vc
   ```

6.1.9.4 Passing Caller Passwords As Null

Use the `allow kerberos null password` configuration parameter to configure the server to allow Kerberos administrators to pass caller passwords.

The caller password is the password for the user executing `alter login .. modify password`.

Set the `allow kerberos null password` configuration parameter to 1 to enable the server to pass caller passwords as null:

```
sp_configure "allow kerberos null password", 1
```

> **i** Note
>
> `allow kerberos null password` does not reset the login password to null. Instead it allows you to pass the caller password as null so you can reset login passwords.

For example, if `user1`, with sso_role and a non-null password of `abc123`, logs into SAP ASE with the `-V` option (Kerberos authentication):

```
isql_r  -U user1 -P abc123 -V -Sbig_server
1> select @@authmech
2> go
 ------------------------------
 kerberos
```

If another login, `user2`, needs to change their password, `user1` can issue this command, using a null password:

```
alter login user2 with password null modify password sybase3
```

6.1.9.5 Using MIT Kerberos Security Services on Windows 64-bit

MIT Kerberos security services are supported on Windows 64-bit.

Context

In order to use the MIT Kerberos Security services on Windows 64bit, changes are required in `libtcl64.cfg`, `libtcl.cfg`, and `sql.ini` files.

Procedure

1. Update the `[SECURITY]` section of the `%SYBASE%\%SYBASE_OCS%\ini\libtcl64.cfg` file.
   a. Add the following to the `[SECURITY]` section:

   ```
   csfkrb5=LIBSYBSKRB64 secbase=@<REALM>
   libgss=<MIT_KRB_64_INSTALL_DIR> \bin\gssapi64.dll
   ```

   > **i Note**
   >
   > For the above example:
   >
   > `<REALM>` should be replaced with the Kerberos realm name.
   >
   > `<MIT_KRB_64_INSTALL_DIR>` should be replaced with the directory where MIT Kerberos version 4.0.1 for Windows 64-bit is installed.
   >
   > The path to the `gssapi` library, used in the `libtcl64.cfg` file, cannot contain whitespaces.

   This change allows 64-bit ASE and 64-bit Open Client applications to use the security driver library `libsybsmssp64.dll` at runtime. The `libsybsmssp64.dll` library is located in `%SYBASE%\%SYBASE_OCS%\dll`, along with other Open Client dynamic link libraries.
2. Update the `[SECURITY]` section in the `%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg` file.
   a. Add the following to the `[SECURITY]` section:

   ```
    csfkrb5=LIBSYBSKRB secbase=@<REALM>
   libgss=<MIT_KRB_32_INSTALL_DIR>    \bin\gssapi32.dll
   ```

   > **i Note**
   >
   > For the above example:
   >
   > `<REALM>` should be replaced with the Kerberos realm name.
   >
   > `<MIT_KRB_32_INSTALL_DIR>` should be replaced with the directory where MIT Kerberos version 4.0.1 for Windows 32-bit is installed.
   >
   > The path to the `gssapi` library, used in the `libtcl64.cfg` file, cannot contain whitespaces.

   The `libtcl.cfg` is used by the 32bit `isql` utility and 32-bit OpenClient applications.
3. Choose one of the following methods to specify the OID value for MIT Kerberos.
   - Update the `[SECMECH]` section of the `%SYBASE%\ini\sql.ini` file. Add the following to the `sql.ini` file:

     ```
     [ASENAME]
     master=TCP,<host>,<port>
     query=TCP,<host>,<port>
     secmech=1.3.6.1.4.1.897.4.6.6
     ```

   - Use the `dsedit` utility to add the 'Server Security' attribute value of `'1.3.6.1.4.1.897.4.6.6'` to your server.

     > **i Note**
     >
     > This OID value derived from the `%SYBASE%\ini\objectid.dat` file which should not be modified

.

4. Make sure Adaptive Server is configured for security services. For example, to enable services with LAN Manager, execute:

   ```
   sp_configure "use security services", 1
   ```

   For more information, see *Using Security Services with NT LAN Manager* in the *Configuration Guide for Windows NT*.

5. Connect to the Adaptive Server without a user name and password. For example:
   - `isql -V -SASENAME`
   - `isql64 -V -SASENAME`

6.1.9.6 Establish Kerberos Security for Remote Connections

SAP ASE acts as the client when it connects to another server to execute a remote procedure call (RPC) and for remote connections through Component Integration services (CIS).

For remote server logins through SAP ASE for RPC execution, one physical connection is established between the two servers. The servers use the physical connection to establish one or more logical connections—one logical connection for each RPC.

SAP ASE supports end-to-end Kerberos authentication for Kerberos logins that attempt remote server connections through CIS using the credential delegation feature provided by Kerberos version 5.

The credential delegation or ticket forwarding allows a Kerberos client to delegate the credential when connecting to a server, thereby allowing the server to initiate Kerberos authentication for further connections to other servers on behalf of Kerberos client.

A Kerberos client connected to SAP ASE can request a Remote Procedure Call (RPC) to SAP ASE, and for general distributed query processing requests to a remote server through CIS by using the Kerberos credential delegation feature. The Kerberos authentication feature used for connections to remote servers is not supported for remote server logins. For information about configuring CIS Kerberos Authentication, see *Configuration for Component Integration Services Remote Procedure Calls*, in the *Component Integration Services User Guide*.

> **i Note**
>
> If the ID of any server gets changed after configuring high availability (HA) with SAP ASE, you may experience issues with Kerberos authentication for remote connections to those servers with IDs that have changed. Restart the SAP ASE server to reconnect.

6.1.9.6.1 Unified Login and the Remote Server Logins

If the local server and remote server are set up to use security services, you can use unified login on both servers.

Use one of these two methods:

- The system security officer defines a user as "trusted" with `sp_remoteoption` on the remote server. The user gains access to the local server using a "unified login" and executes an RPC on the remote server. The user is trusted on the remote server and does not need to supply a password.
- A user specifies a password for the remote server when he or she connects to the local server. The facility to specify a remote server password is provided by the `ct_remote_pwd` routine available with Open Client Client-Library/C. See the *Open Client Client-Library/C Reference Manual*.

6.1.9.6.2 Obtain Information About Remote Servers

`sp_helpserver` displays information about servers.

When you run `sp_helpserver` without an argument, it provides information about all the servers listed in `sysservers`. You can specify a particular server to receive information about that server. The syntax is:

```
sp_helpserver [<server>]
```

For example, to display information about the GATEWAY server, execute:

```
sp_helpserver GATEWAY
```

6.1.10 Principal Names

The principal name is the name the server uses to authenticate with the Kerberos key distribution center (KDC).

When you have multiple instances running, you must have different principal names for each server.

6.1.10.1 Server Principal Name

Use the DSLISTEN and DSQUERY environment variables, or the `dataserver -s<server_name>` command line option to specify the server name.

Use either the `setenv` command or the `-k dataserver` option to set the principal name.

By default, the principal name is the name of the SAP ASE server. To specify a different name, set SYBASE_PRINCIPAL before starting SAP ASE to use Kerberos:

```
setenv SYBASE_PRINCIPAL <<name of principal>>
```

Once you have set a principal name, the value of this variable is used to authenticate itself to Kerberos.

To specify a principal name when starting the server, use:

```
-k <<server principal name>>
```

When you start a server with the Kerberos security mechanism enabled, the principal name specified with the `-k` option is used for Kerberos authentication. If the `-k` option is not specified, the principal name in the environment variable SYBASE_PRINCIPAL is used. If neither is specified, the server name is used for authentication.

Kerberos Open Client connections that use different server principal names are accepted if the entry for the principal name is present in the `keytab` file. To allow connections with different principal names:

- Pass an empty string as a parameter for the `-k` option, or
- Set the SYBASE_PRINCIPAL environment variable to "". For example:

  ```
  export SYBASE_PRINCIPAL=""
  ```

Example

In this example, the server name is "secure_ase" and the realm name is "MYREALM.COM." The server name is specified on the command line with `-s` parameter to the `dataserver`. The current realm is specified in `libtcl.cfg` by a `secbase` attribute value:

```
[SECURITY]
csfkrb5=libskrb.so libgss=/krb5/lib/libgss.so secbase=@MYREALM.COM
```

The default principal name is "secure_ase@MYREALM.COM." If the principal name defined in the `keytab` file is "aseprincipal@MYREALM.COM," you can override the default principal name by setting a server principal name using options 1 or 2 below:

- Option 1, specify `-k ''`:

  ```
  %
  $SYBASE/$SYBASE_ASE/bin/dataserver -dmaster.dat
  -s secure_ase -k aseprincipal@MYREALM.COM
  ```

  The principal name used to authenticate with Kerberos is "aseprincipal@MYREALM.COM."
- Option 2, set SYBASE_PRINCIPAL:

  ```
  setenv SYBASE_PRINCIPAL aseprincipal@MYREALM.COM
  $SYBASE/$SYBASE_ASE/bin/dataserver -dmaster.dat
  -s secure_ase
  ```

  The principal name used to authenticate with Kerberos is "aseprincipal@MYREALM.COM," the value of `$SYBASE_PRINCIPAL`.
- Option 3, neither `-k` nor SYBASE_PRINCIPAL is set:

  ```
  % $SYBASE/$SYBASE_ASE/bin/dataserver -dmaster.dat
  -s secure_ase
  ```

  The principal name used to authenticate with Kerberos is "secure_ase@MYREALM.COM."

6.1.10.2 Use sybmapname to Handle User Principal Names

`sybmapname` converts external user principal names used in the Kerberos environment to the namespace of user logins.

You can customize the `sybmapname` shared object and map names specified in the Kerberos input buffer to names suitable for a login to the server output buffer.

Use the `sybmapname` shared object to perform the custom mapping between the user principal name and the login name. This shared object is optionally loaded at server start-up, and the function `syb_map_name` contained in the shared object is called after a successful Kerberos authentication and just before the user principal is mapped to a login in the `syslogins` table. This function is useful when the user principal name and the login name to be mapped are not identical.

```
syb__map_name(NAMEMAPTYPE *protocol, char *orig,
int origlen, char *mapped, int *mappedlen)
```

where:

- `NAMEMAPTYPE *protocol` – refers to a structure reserved for usage of this function.
- `char *orig` – is an input buffer that is not null-terminated.
- `int origlen` – is the input buffer length, which should be less than or equal to 255 characters.
- `char *mapped` – is an output buffer that should not be null-terminated.
- `int *mappedlen` – is an output buffer length, which should be less than or equal to 30.

`syb__map_name` returns a value greater than 0 if the mapping succeeds, or returns a value of 0 if no mapping occurred, and it returns a value less than 0 when an error occurs in `syb__map_name`. When an error occurs, reporting the mapping failure is written to the error log.

For example, to authenticate a Kerberos user on SAP ASE:

1. Configure SAP ASE to use the Kerberos security mechanism.
   A sample `sybmapname.c` file is located in `$SYBASE/$SYBASE_ASE/sample/server/sybmapname.c`.
2. Modify `sybmapname.c` to implement your logic. See *Precautions When Using sybmapname*.
3. Build the shared object or DLL using the generic platform-specific `makefile` supplied. You may need to modify the `makefile` to suit your platform-specific settings.
4. Place the resulting shared object generated in a location specified in your $LD_LIBRARY_PATH on UNIX machines, and PATH variable on Windows machines. The file should have read and execute permissions for the “sybase” user.

> **Note**
>
> SAP recommends that only the “sybase” user is allowed read and execute permissions, and that all other access should be denied.

Precautions When Using sybmapname

When coding for `sybmapname`:

- Use caution when making modifications to the sample `sybmapname.c` program. Avoid using code that may create a segmentation fault, that may call `exit`, that may call `system calls`, that may change UNIX signals, or that makes any blocking calls. Improper coding or calls may interfere with the SAP ASE server engine.

  > **i Note**
  >
  > SAP bears no responsibility for coding errors in `sybmapname`.

- Code defensively, check all pointers before dereferencing them, and avoid system calls. The functions you write must be quick name-filtering functions.
- Do not use `goto` statements since, depending on the platform, they may cause unexpected side effects.
- If you use multiple realms, use caution when mapping the user principal names to a suitable login name to reflect the realm information. For example, if you have two users whose user principal names are `userA@REALMONE` and `userB@REALMTWO`, respectively, map them to the login names `userA_REALMONE` and `userB_REALMTWO`, instead of `userA` or `userB`. This distinguishes the two users who belong to different realms.

Verifying Your login to SAP ASE Using Kerberos Authentication

To verify your login to SAP ASE using Kerberos authentication, assume that:

- `$SYBASE` refers to your release and installation directory.
- `$SYBASE_ASE` refers to the SAP ASE version directory that contains your server binary.
- `$SYBASE_OCS` refers to the Open Client/Server version directory.

Example: Example 1

If a client's principal name is `user@REALM`, and the corresponding entry in `syslogins` table is `user_REALM`, you can code `sybmapname` to accept the input string `user@realm` and to convert the input string to the output string `user_REALM`.

Example: Example 2

`sybmapname` is loaded by SAP ASE at runtime and uses its logic to do the necessary mapping.

The following actions and output illustrate the `sybmapname` function. The `sybmapname.c` file containing the customized definition for `syb__map_name()` should be compiled and built as a shared object (or DLL), and finally placed in the appropriate path location. Start SAP ASE with the Kerberos security mechanism enabled.

To initialize the Ticket Granted Ticket (TGT), which is a encrypted file that provides identification:

```
$ /krb5/bin/kinit johnd@public
Password for johnd@public:
$
```

To list the TGT:

```
$ /krb5/bin/klist
   Cache Type: Kerberos V5 credentials cache
   Cache Name: /krb5/tmp/cc/krb5cc_9781
Default principal: johnd@public
```

Log in as "sa" and verify the user login for "johnd":

```
$ $SYBASE/$SYBASE_OCS/bin/isql -Usa -P
      -Ipwd`/interfaces
1>
1> sp_displaylogin johnd
2> go
No login with the specified name exists.
(return status = 1)
1> sp_displaylogin JOHND
2> go
Suid: 4
Loginame: JOHND
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ANY
(return status = 0)
```

Successful Kerberos authentication, maps lower-case `johnd` to uppercase `JOHND` using the `sybmapname` utility, and allows user `johnd` to log in to SAP ASE:

```
$ $SYBASE/$SYBASE_OCS/bin/isql -V -I'pwd'/interfaces
1>
```

6.2 Configure SAP ASE for LDAP User Authentication

The LDAP user authentication allows client applications to send user name and password information to SAP ASE for authentication by the LDAP server instead of `syslogins`.

Authentication using the LDAP server allows you to use server-wide passwords instead of SAP ASE or application-specific passwords.

LDAP user authentication is ideal if you want to simplify and centralize user administration, or want to avoid unnecessary complexities for user administration.

LDAP user authentication works with directory servers that meet Version 3 of the LDAP protocol standard, including Active Directory, iPlanet, and OpenLDAP Directory Server.

Use one of these authentication algorithms with LDAP user authentication:

- Composed DN for authentication, available for SAP ASE version 12.5.1 or later, or,
- Searched DN for authentication, available for SAP ASE version 12.5.2 and later.

These algorithms differ in how they obtain a user's distinguished name (DN).

The primary data structure used with the LDAP protocol is the LDAP URL.

An LDAP URL specifies a set of objects or values on an LDAP server. SAP ASE uses LDAP URLs to specify an LDAP server and search criteria to use to authenticate login requests.

The LDAP URL uses this syntax:

```
ldapurl::=ldap://host:port/node/attributes [base | one | sub] filter
```

where:

- `<host>` – is the host name of the LDAP server.
- `<port>` – is the port number of the LDAP server.
- `<node>` – specifies the node in the object hierarchy at which to start the search.
- `<attributes>` – is a list of attributes to return in the result set. Each LDAP server may support a different list of attributes.
- `base | one | sub` – qualifies the search criteria. `base` specifies a search of the base node; `one` specifies a search of the base node and one sublevel below the base node; `sub` specifies a search of the base node and all node sublevels.
- `filter` – specifies the attribute or attributes to be authenticated. The filter can be simple, such as `uid=*`, or compound, such as `(uid=*)(ou=group)`.

6.2.1 Composed DN Algorithm

When you use the composed DN algorithm, this is login sequence used.

1. Open Client connects to an listener port.
2. The listener accepts the connection.
3. Open Client sends an internal login record.
4. The SAP ASE server reads the login record..
5. SAP ASE binds to the LDAP server with a DN composed from the primary URL and the login name from the login record. This bind also uses the password from the login record.
6. The LDAP server authenticates the user, returning either a success or failure message.
7. If the Primary URL specifies a search, then SAP ASE sends the search request to the LDAP server.
8. The LDAP server returns the results of the search.
9. The login is accepted or rejected based on the search results.

6.2.2 Searched DN Algorithm

When you use the searched DN algorithm, this is the login sequence used.

1. Open Client connects to an SAP ASE listener port.
2. The SAP ASE listener accepts the connection.
3. Open Client sends an internal login record.
4. SAP ASE reads the login record.
5. SAP ASE binds to the LDAP server with a directory server access account.
   The connection established in steps 5 and 6 may persist between authentication attempts from SAP ASE to reuse connections to DN searches.
6. The LDAP server authenticates the user, returning either a success or failure message.
7. SAP ASE sends search requests to the LDAP server based on the login name from the login record and the DN lookup URL.
8. The LDAP server returns the results of the search.
9. SAP ASE reads the results to obtain an a value of attribute from the DN lookup URL.
10. SAP ASE uses the value of attribute as the DN and the password from the login record to bind to the LDAP server.
11. The LDAP server authenticates the user, returning either a success or failure message.
12. If the primary URL specifies a search, SAP ASE sends the search request to the LDAP server.
13. The LDAP server returns the results of the search.
14. SAP ASE accepts or rejects the login, based on the search results.

SAP ASE reports a generic login failure to the client if any of these authentication criteria are not met.

You may skip steps 12 and 13 by not specifying search criteria in the primary or secondary URL strings. The authentication completes, displaying the success or failure returned by step 11.

6.2.3 Configuring LDAP in New SAP ASE Installations

You can configure SAP ASE for LDAP authentication.

Procedure

1. Specify the LDAP URL search strings and access account values.
2. Set `enable ldap user auth` to 2.
3. Add users in the LDAP directory server using LDAP vendor-supplied tools.
4. Add users using `create login`. You can also use `sp_maplogin` to automatically create login accounts upon authentication or apply other login controls.

6.2.4 Migrating existing SAP ASE Servers to LDAP

To avoid disruption of service in existing server installations, migrate SAP ASE to LDAP.

Procedure

1. Specify an LDAP URL search string to SAP ASE.
2. Set the configuration parameter `enable ldap user auth` to 1.
3. Add users in the LDAP directory server.
4. When all users are added to the LDAP server, set `enable ldap user auth` to 2 to require all authentications to be performed with LDAP, or use `sp_maplogin` to override configuration parameters with login controls.

6.2.5 LDAP User Authentication Administration

Use `sp_ldapadmin` to create or list an LDAP URL search string, verify an LDAP URL search string or login, and specify the access accounts and tunable LDAP user authentication (LDAPUA) related parameters.

You must have the SSO role to execute `sp_ldapadmin`.

Composed DN Examples

If you use a simple LDAP server topology and schema, you can use a composed DN algorithm for user authentication. If you use commercially available schemas (for example, iPlanet Directory Servers or OpenLDAP Directory Servers), users are created as objects in the same container in the LDAP server tree, and SAP ASE determines the user's DN from the object's location. However, there are restrictions on the LDAP server's schema:

- You must specify the filter with the attribute name that uniquely identifies the user to be authenticated.
- You must specify the filter with the attribute `name=*`. The asterisk is a wildcard character. The appropriate attribute name to use in the filter depends on the schema used by the LDAP server.
- The login name is the same as the short user name for example, a UNIX user name.
- The DN uses the short user name rather than a full name with embedded spaces or punctuation. For example, `jqpublic` meets the restriction for a DN, but "John Q. Public" does not.

iPlanet Example

LDAP vendors may use different object names, schema, and attributes than those used in these examples. There are many possible LDAP URL search strings, and valid sites may also extend schemas locally or use them in ways that differ from each other:

- This example uses the `uid=*` filter. To compose the DN, SAP ASE replaces the wildcard with the login name to be authenticated, and appends the resulting filter to the node parameter in the LDAP URL. The resulting DN is:

  ```
  uid=myloginname,ou=People,dc=mycomany,dc=com
  ```

- After a successful bind operation, the connection is used to search for attribute names, such as `uid`, that are equal to the login name:

  ```
  sp_ldapadmin set_primary_url, 'ldap://myhost:389/
  ou=People,dc=mycompany,dc=com??sub?uid=*'
  ```

- This example uses the schema defined in OpenLDAP 2.0.25, with an attribute name of `cn`.
  The composed DN is `cn=myloginname,dc=mycompany,dc=com`:

  ```
  sp_ldapadmin set_primary_url, 'ldap://myhost:389/dc=mycompany,dc=com??sub?
  cn=*'
  ```

Searched DN Examples

Use the searched DN to use an Active Directory server or other LDAP server environment that does not meet the restrictions to use the composed DN algorithm.

- Perform these steps for an Active Directory server using a commercially available user schema from a Windows 2000 Server.
  1. Set the access account information:

     ```
     sp_ldapadmin set_access_acct,
     'cn=Admin Account, cn=Users, dc=mycompany, dc=com',
     'Admin Account secret password'
     ```

  2. Set the primary URL:

     ```
     sp_ldapadmin set_primary_url, 'ldap://hostname:389/
     ```

  3. Set the DN lookup URL search string:

     ```
     sp_ldapadmin set_dn_lookup_url, 'ldap://hostname:389/
     cn=Users,dc=mycompany,dc=com?distinguishedName?one?samaccountname=*'
     ```

On Windows, the short name is typically referred to as the "User Logon Name" and is given the attribute name `samaccountname` in the default schema. This is the attribute name used to match the login name. The DN for a user contains a full name with punctuation and embedded spaces (for example, `cn=John Q. Public, cn=Users, dc=mycomany, dc=com`. The DN on Windows does not use the short name, so the searched DN algorithm is appropriate for sites using the Active Directory schema (the default) as the LDAP server. The primary URL does not specify a search. Instead, it relies on the bind operation for authentication.

Examples Using Search Filters to Restrict Access

You can use LDAP URL search strings to restrict access to groups of users on LDAP servers. For example, to restrict logins to users in an accounting group. use a compound filter to restrict access to the group of users where attribute `group=accounting`.

- The following LDAP URL string uses the composed DN algorithm for an iPlanet server:

  ```
  sp_ldapadmin set_primary_url, 'ldap://myhost:389/ou=People,dc=mycompany,
  dc=com??sub?(&(uid=*)(group=accounting))'
  ```

  SAP ASE binds with DN `uid=mylogin,ou=People,dc=mycompany,dc=com`. After successfully binding with this identity, it searches for:

  ```
  "ou=People,dc=mycompany,dc=com??sub?(&(uid=mylogin)(group=accounting))"
  ```

  Authentication succeeds if this search returns any objects.
- These examples use LDAP URL strings with compound filters:

  ```
  sp_ldapadmin set_primary_url, 'ldap://myhost:389/
  ou=people,dc=mycompany,dc=com??sub?(&(uid=*)(ou=accounting)  (l=Santa Clara))'
  ```

  ```
  sp_ldapadmin, set_primary_url, 'ldap://myhost:389/
  ou=people,dc=mycompany,dc=com??sub?(&(uid=*)(ou=Human%20Resources))'
  ```

6.2.5.1 LDAP User Authentication Password Information Changes

There are two LDAP user authentication-related informational messages that SAP ASE obtains from the LDAP server and passes to the client.

- If you log in to an SAP ASE server using an LDAP authentication mechanism with an LDAP user authentication password that is about to expire, you see:

  ```
  Your password will expire in <<number>> days.
  ```
- If you attempt to log in to SAP ASE using an LDAP authentication mechanism after the LDAP server administrator resets your password or after your LDAP server password has expired, you see message 4002:

  ```
  Login failed
  ```

  If auditing is enabled and the `errors` auditing option is turned on, message 4099 is sent to the audit log:

  ```
  Your LDAP password has expired.
  ```

> **i Note**
>
> Configure your LDAP server to give this additional information. Additionally, SAP ASE must support the transmission of LDAP password controls to an LDAP client.

6.2.5.2 Failover Support

When a major failure occurs in the LDAP directory server specified by the primary URL, and the server no longer responds to network requests, SAP ASE attempts to connect to the secondary LDAP directory server specified by the secondary URL.

SAP ASE uses the LDAP function `ldap_init` to determine if it can open a connection to the LDAP directory server. A null or invalid primary URL string causes an SAP ASE server to attempt to fail over to a secondary URL. Failures returned by LDAP bind or search operations do not cause a fail over to the secondary URL.

6.2.6 Logins and LDAP User Accounts

Once you enable LDAP user authentication, choose and set an authentication algorithm and URL strings, you must configure the user accounts.

The LDAP administrator creates and maintain accounts in the LDAP server, and the database administrator creates and maintains accounts in SAP ASE. Alternatively, the database administrator can choose administration options that allow flexibility with login accounts when integrating SAP ASE with external authentication mechanisms such as LDAP server. The database administrator continues to administer the account roles, default database, default language, and other login-specific attributes using traditional commands and procedures.

Updates to syslogins from LDAP describes the updates that SAP ASE makes at login time to the `syslogins` table. These updates assume that LDAP user authentication is configured, the login is not restricted from using LDAP, and you have not set the `create login` mapping.

These are the updates to syslogins from LDAP:

Does the row exist in syslogins?	LDAP server authentication succeeds?	Changes in syslogins
No	Yes	No change, login fails
No	No	No change, login fails
Yes	Yes	Update row if password has changed
Yes	No	No change

6.2.7 Secondary Lookup Server Support

Uninterrupted support is provide for clients that are authenticated by an LDAP server. You can specify a secondary LDAP lookup server to fail over from a primary LDAP server in the event of the LDAP server failure or planned downtime.

The health of the URL set is monitored through the following states:

- INITIAL – indicates that LDAP user authentication is not configured.
- RESET – indicates that the URL has been entered with administrative commands.

- READY – indicates that the URL is ready to accept connections.
- ACTIVE – indicates that the URL has performed a successful LDAP user authentication.
- FAILED – indicates that there is a problem connecting to the LDAP server.
- SUSPENDED – indicates that the URL is in maintenance mode, and will not be used.

The following sequence of events describe the failover and manual failback:

1. The primary and secondary URL sets are configured and in a READY state.
2. The connections are authenticated using the primary server infrastructure.
3. The primary server fails, and its state is changed to FAILED.
4. Connections automatically begin authentication through the secondary server infrastructure.
5. The primary server is repaired and brought back online by an LDAP administrator. The primary LDAP server state is changed by an administrator to READY.
6. New connections are authenticated using the primary server.

> **i Note**
>
> Once the SAP ASE server has failed over to the secondary LDAP server, a database administrator must manually activate the primary LDAP server before it can be used again.

When SAP ASE encounters errors connecting to an LDAP server, it retries the authentication three times. If the errors persist, the LDAP server is marked as FAILED.

Use `sp_ldapadmin` to configure secondary lookup LDAP servers.

- To set the secondary DN lookup URL, enter:
  `sp_ldapadmin set_secondary_dn_lookup_url, <URL>`
- To set the administrative access account for the secondary DN lookup URL, enter:

  ```
  sp_ldapadmin set_secondary_access_acct, <DN>, <password>
  ```
- To suspend the use of a primary or secondary URL for authentication, enter:
  `sp_ldapadmin suspend, {primary | secondary}`
- To activate the set of primary or secondary URLs for authentication, enter:
  `sp_ldapadmin activate, {primary | secondary}`
- To display details about the primary and secondary LDAP server settings and status, enter:
  `sp_ldapadmin list`
  `sp_ldapadmin list` combines previous outputs from `list_access_acct` and `list_urls`. It has the following expected output for the primary and secondary servers:
  - Search URL
  - Distinguished name lookup URL
  - Access account DN
  - Active [true | false]
  - Status [ready | active | failed | suspended | reset]
- To display DN lookup URLs for the secondary server, enter:
  `sp_ldapadmin list_urls`
- To display the administrative account for the secondary DN lookup URL, enter:
  `sp_ldapadmin list_access_acct`
- To display subcommands, enter:
  `sp_ldapadmin help`

6.2.8 LDAP Server State Transitions

Use the `sp_ldapadmin` command to show LDAP server state transitions.

Table 7: State Transitions When sp_ldapadmin set_URL is Executed

Initial state	Final state
INITIAL	RESET
RESET	RESET
READY	READY
ACTIVE	RESET
FAILED	RESET
SUSPENDED	RESET

`set_URL` represents one of these commands

- `set_dn_lookup_url`
- `set_primary_url`
- `set_secondary_dn_lookup_url`
- `set_secondary_url`

Table 8: State Transitions When sp_ldapadmin Suspend is Executed

Initial state	Final state
INITIAL	Error
RESET	SUSPENDED
READY	SUSPENDED
ACTIVE	SUSPENDED
FAILED	SUSPENDED
SUSPENDED	SUSPENDED

Table 9: State Transitions When sp_ldapadmin Activate is Executed

Initial state	Final state
INITIAL	Error
RESET	READY
READY	READY

Initial state	Final state
ACTIVE	ACTIVE
FAILED	READY
SUSPENDED	READY

The following tables show the LDAP server state transitions carried out implicitly by SAP ASE.

Table 10: State Transitions When SAP ASE is Restarted

Initial state	Final state
INITIAL	INITIAL
RESET	RESET
READY	READY
ACTIVE	READY
FAILED	FAILED
SUSPENDED	SUSPENDED

SAP ASE only attempts an LDAP login if the LDAP server is in a READY or ACTIVE state.

Table 11: State Transitions When an LDAP Login Succeeds

Initial state	Final state
READY	ACTIVE
ACTIVE	ACTIVE

Table 12: State Transitions When an LDAP Login Fails

Initial state	Final state
READY	FAILED
ACTIVE	FAILED

6.2.9 LDAP User Authentication Tuning

Configure and tune SAP ASE options based on the load of incoming connections and the SAP ASE/LDAP server infrastructure.

Configure these options based on the number of simultaneous incoming requests:

- Use `sp_configure` to set `max native threads`, which indicates the number of native threads per engine.
- Use `sp_ldapadmin` to configure `max_ldapua_native_threads`, which indicates the number of LDAP user authentication native threads per engine.

Configure the `set_timeout` option (which indicates the LDAP server bind and search timeouts) based on the network and the health of the SAP ASE/LDAP server infrastructure.

Configure the `set_abandon_ldapua_when_full` option to specify the behavior when incoming connections have consumed `max_ldapua_native_threads`:

Use these `sp_ldapadmin` options to configure the LDAP server for better performance:

- `set_max_ldapua_desc` – manages the concurrency of the LDAPUA connection requests. If you are using a distinguished name algorithm, setting `set_max_ldapua_desc` to a larger number expedites the LDAPUA connections SAP ASE is processing.
- `set_num_retries` – sets the number of attempts. Tune this number according to the number of transient errors between SAP ASE and the LDAP server. You can nullify transient errors by configuring the number of retries.
- `set_log_interval` – controls the number of messages sent to the error log for diagnostic purposes. Using a low number clutters the error log may be helpful in identifying specific errors. Using a large number sends fewer messages to the error log, but does not have the same investigative value. Tune `set_log_interval` according to your error log size.

6.2.10 Add Tighter Controls on Login Mapping

Use `sp_maplogin` to map users that are authenticated with LDAP or PAM to the local login.

> **i Note**
>
> To map a user authenticated with Kerberos, use `sybmapname` instead of `sp_maplogin`.

Only users with `sso_role` can create or modify login mappings using `sp_maplogin`.

SAP ASE avoids conflicts between an authentication mechanism setting for a login and a mapping that uses the login. Potential mapping conflicts are detected by the stored procedure `sp_maplogin` or the commands `alter login`, or `create login`.

These controls do not allow maps:

- From one login name to another login name
- From an external name that already exists as a local login
- To a nonexistent login name

Additionally, when the authentication mechanism is specified with a mapping, the mechanism is checked with the authentication mechanism set in the target login.

If a target login's authentication mechanism restricts the login to use a particular authentication mechanism, then the mechanism specified with the mapping must match either that specified for the login or match the "ANY" authentication mechanism.

When `sp_maplogin` detects that a conflict exists, `sp_maplogin` fails and reports an error that identifies the conflict.

Similarly, `alter login` and `create login` check for an existing mapping that may conflict with the `authenticate with` option for the user login. When `alter login` or `create login` detect a conflict, an error is reported to identify any conflicts with a login mapping.

Example: Example 1

Maps an LDAP user to the "sa" login. A company has adopted LDAP as their repository for all user accounts and has a security policy that requires LDAP authentication of all users including database administrators, "adminA" and "adminB," who may manage hundreds of servers. Auditing is enabled, and login events are recorded in the audit trail.

To map these administrator accounts to "sa," enter:

```
sp_maplogin LDAP, 'adminA', 'sa'
go
sp_maplogin LDAP, 'adminB', 'sa'
go
```

Require all users to authenticate using LDAP authentication:

```
sp_configure 'enable ldap user auth', 2
go
```

When "adminA" authenticates during login to a server, the distinguished name associated with "adminA" rather than only "sa" is recorded in the login audit event. This allows each individual performing an action to be identified in the audit trail.

Because the "adminA" and "adminB" password is set in the LDAP server, there is no need to maintain the "sa" password on all servers being managed.

This example also allows different external identities and passwords to be used for authentication, while their actions within the server still require the special privileges associated with "sa" account.

Example: Example 2

Uses both PAM and LDAP to map users to application logins. A company has adopted both PAM and LDAP authentication but for different purposes. The company security policy defines LDAP as the authentication mechanism for general user accounts, and PAM for special users, such as for a middle-tier application. A middle-tier application may establish a pool of connections to SAP ASE to handle requests on behalf of users of the middle-tier application.

Configure SAP ASE for both LDAP and PAM user authentication:

```
sp_configure 'enable ldap user auth', 2
go
sp_configure 'enable pam user auth', 2
go
```

Establish a login `appX` locally with permissions that are appropriate for the middle-tier application:

```
create login appX with password myPassword
go
alter login appX authenticate with PAM
go
```

Instead of hard-coding a simple password in "appX" and maintaining the password consistently in several different servers, develop a custom PAM module to authenticate the application in a centralized repository using additional facts to verify the middle-tier application.

Client application login "appY" requires LDAP authentication of the user with its LDAP identity and password. Use `sp_maplogin` to map all LDAP authenticated users to login "appY,"

```
create login appY with password myPassword
go
sp_maplogin LDAP, NULL, 'appY'
go
```

Users of "appY" are authenticated with their company identity and password, then mapped to a local login "appY" to execute database actions. Authentication has occurred with the identity of the LDAP user, which is recorded in the audit trail, and executes with permissions appropriate to the application login "appY."

6.2.10.1 Login Mapping of External Authentication

When you configure an external authentication mechanism, if there is a single mapping of an external user to an internal login, and if the mapping is successfully authenticated, the internal login password is updated to match the external user's password.

For example:

1. A user has an login name of `user_ase` (with password `user_password`), and an LDAP login name of `user_ldap` (with password `user_ldappasswd`).
   The produces a one to one mapping for `user_ldap` to `user_ase`.
2. When `user_ldap` logs into the server using the `user_ldappassword`, the password for `user_ase` is updated to `user_ldappassword`

The benefit of mapping the login name to the LDAP password is that the user can log in with the most recently used LDAP password if the LDAP server crashes. That is, when a user has a one-to-one mapping of a user name to an LDAP password for authentication, the user appears to have uninterrupted authentication because the password is updated locally when it is used to authenticate the login.

However, the password is not updated locally when more than one external user is mapped to the local user. If the LDAP server crashes, multiple external users cannot be mapped to a single user.

6.2.11 Troubleshoot LDAP User Authentication Errors

SAP ASE may experience the following transient errors when communicating with the LDAP server.

These errors are generally resolved by retrying the connection. If the errors persist after three retry attempts, SAP ASE marks the LDAP server as FAILED.

- LDAP_BUSY – server is busy.
- LDAP_CONNECT_ERROR – error during a connection.
- LDAP_LOCAL_ERROR – error on the client side.
- LDAP_NO_MEMORY – cannot allocate memory on the client side.
- LDAP_OPERATIONS_ERROR – error on the server side.
- LDAP_OTHER – unknown error code.
- LDAP_ADMINLIMIT_EXCEEDED – a search has exceeded a limit.
- LDAP_UNAVAILABLE – server cannot process the request.
- LDAP_UNWILLING_TO_PERFORM – server is not going to process the request.
- LDAP_LOOP_DETECT – a loop has been detected during a referral.
- LDAP_SERVER_DOWN – server is not reachable (connection fails).
- LDAP_TIMEOUT – LDAP API fails because operation does not complete in the user-specified amount of time.

Transient errors and a large number of simultaneous login requests may lead to a large number of repeated error messages in the error log. To increase the readability of the log, this error message logging algorithm is used:

1. If a message is being logged for the first time, log it.
2. If the last time the message was logged was greater than 3 minutes:
    - Log the error message.
    - Log the number of times the message was repeated since the message was last printed.
    - Log the time elapsed, in minutes, since the message was printed.

Authentication failures arising from the following are not considered LDAP errors and are not conditions for retrying the authentication request:

- Bind failure due to bad password or an invalid distinguished name.
- A search after a successful bind that returns a result set of 0 or no attribute value.

Syntax errors found while parsing the URL are caught when an LDAP URL is set, and therefore do not fall into any of the above categories.

6.2.12 Configuring an LDAP Server

User authentication for Lightweight Directory Access Protocol (LDAP) supports the Secure Sockets Layer/ Transport Layer Security (SSL/TLS) protocol, providing secure data transmission between SAP ASE and an LDAP server.

Procedure

1. Make sure that all trusted root certificates are located in the same file.

   After you define the trusted servers, a secure connection is configured, where `<servername>` is the name of the current server. If you:

   - Have defined `$SYBASE_CERTDIR`, certificates are loaded from `$SYBASE_CERTDIR/servername.txt` (for UNIX) or `%SYBASE_CERTDIR%\servername.txt` (for Windows).
   - Have not defined `$SYBASE_CERTDIR`, certificates are loaded from `$SYBASE/$SYBASE_ASE/certificates/servername.txt` (for UNIX) or `%SYBASE%\%SYBASE_ASE%\certificates\servername.txt` (for Windows).
2. Restart the server to change the trusted root certificate file.
3. Use `sp_ldapadmin`, specifying `ldaps://` URLs instead of `ldap://` URLs, to establish a secure connection to a secure port of the LDAP server.
4. Establish a TLS session over a plain TCP connection:

   ```
   sp_ldapadmin 'starttls_on_primary', {<true> | <false>}
   ```

   or

   ```
   sp_ldapadmin 'starttls_on_secondary', {<true> | <false>}
   ```

Results

> **Note**
>
> LDAP server connections do not have a `connect timeout` option; if the LDAP server stops responding, all login connections also stop responding.

6.3 LDAPS User Authentication

The command, `reinit_descriptors` unbinds the LDAP server descriptors and reinitializes the user authentication subsystem.

- This command requires System Security Officer permissions.
- If the Certifying Authority (CA) trusted root file is modified without execution of this command by a user with System Security Officer permissions, the housekeeping utility chores task uses a new chore, designed to reinitialize the user authentication subsystem every 60 minutes.
  SAP ASE supports modifications to the trusted root file, so that restarting the server is unnecessary.

6.4 Automatic LDAP User Authentication and Failback

SAP ASE provides support for a secondary LDAP server.

The `set_failback_interval` option in `sp_ldapadmin set_failback_interval` sets the interval between attempts to activate failed LDAP servers; if you do not set this parameter, the default value is 15 minutes. See `sp_ldapadmin` in the *Reference Manual: Procedures*.

If the primary URL is marked FAILED, the housekeeper task attempts to activate it, using the primary access account distinguished name (DN) and password. If you have not configured a primary access account, the housekeeper task attempts to use an anonymous bind. If the bind operation fails on the first attempt, the housekeeper task retries the bind operation for the number of retry times configured. If the bind operation succeeds, the primary URL is marked READY.

If the secondary URL is marked FAILED, the housekeeper task attempts to activate the secondary URL in a similar way.

The `reinit_descriptors` option in `sp_ldapadmin` executes when the certificate file is modified, in which case it reinitializes the LDAP user authentication subsystem every 60 minutes.

After you set the failback interval, the housekeeper task checks for failed LDAP servers each time it sweeps through its chores. When it finds a failed LDAP server, it attempts to activate the LDAP server when the failback time interval expires.

6.4.1 LDAP Failback Time Interval

To activate an LDAP server automatically, use `'set_failback_interval'`.

The syntax is:

```
sp_ldapadmin 'set_failback_interval', <time_in_minutes>
```

where:

`<time_in_minutes>` is the value from -1 to 1440 minutes (24 hours):

- A value of 0 indicates that failing back is manual. That is, the housekeeper task does not attempt to automatically fails back the LDAP server. You must perform this task manually.
- A value of -1 sets the fail over time interval to 15 minutes, the default.
- If you issue `sp_ldapadmin 'set_failback_interval'` without any parameters, `sp_ldapadmin` displays the value to which the fail back interval is set.
- If you issue `sp_ldapadmin` without any parameters, `sp_ldapadmin` includes the failback time interval in the output:
- ```
  sp_ldapadmin
  ----------------
  Primary:
      URL:                  ''
      DN Lookup URL:        ''
      Access Account:       ''
      Active:               'FALSE'
      Status:               'NOT SET
      StartTLS on Primary LDAP URL: 'TRUE'
  Secondary:
      URL:                  ''
      DN Lookup URL:        ''
      Access Account:       ''
      Active:               'FALSE'
      Status:               'NOT SET'
      StartTLS on Secondary LDAP URL: 'FALSE'
  Timeout value:            '-1'(10000) milliseconds
  Log interval:             '3' minutes
  Number of retries:        '3'
  Maximum LDAPUA native threads per Engine: '49'
  Maximum LDAPUA descriptors per Engine: '20'
  Abandon LDAP user authentication when full: 'false'
  Failback interval:        '-1'(15) minutes
  (return status = 0)
  ```

This example sets the LDAP failback time interval to 60 minutes:

```
sp_ldapadmin 'set_failback_interval' 60
```

This example sets the LDAP failback

time interval to the default, 15 minutes:

```
sp_ldapadmin 'set_failback_interval' -1
```

This example displays the value to which the failback interval is set:

```
sp_ldapadmin 'set_failback_interval'
```

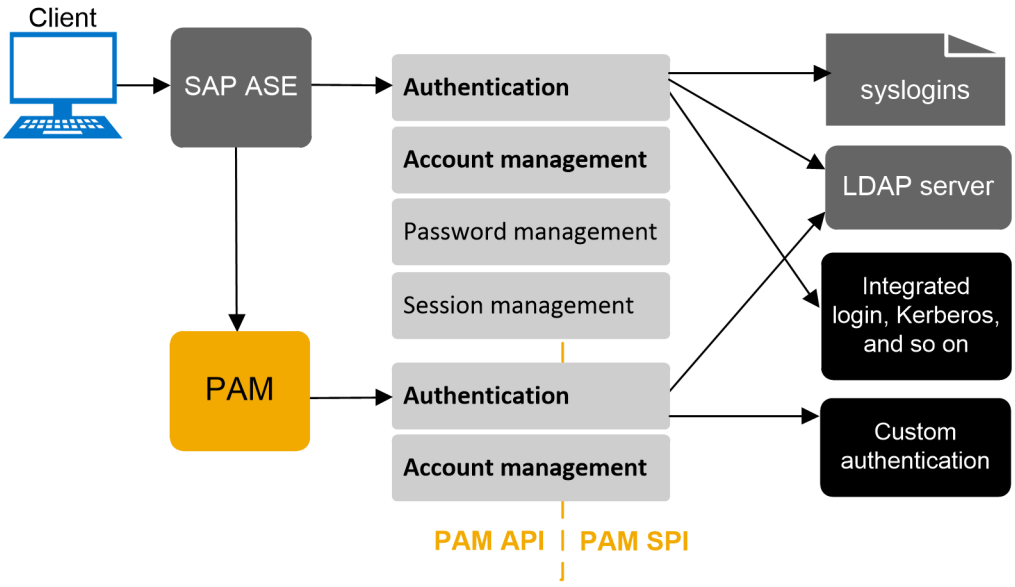
```
The LDAP property 'set_failback_interval' is set to '15
    minutes'.
```
```

6.5 Configure SAP ASE for Authentication Using PAM

Pluggable Authentication Module (PAM) support allows multiple authentication service modules to be stacked and made available without modifying the applications that require authentication.

PAM integrates SAP ASE with Solaris and Linux operating systems and simplifies the management and administration of user accounts and authentication mechanisms, thus reducing the total cost of ownership. Users can customize or write their own authentication and authorization modules.

> **i Note**
>
> PAM support is currently available on Linux and on Solaris platforms. For more information on PAM user authentication, see your operating system documentation.



SAP ASE passes the login name and credentials obtained from the login packet to the PAM API. PAM loads a service provider module as specified in the operating system configuration files and calls appropriate functions to complete the authentication process.

6.5.1 Enable PAM in SAP ASE

Both Linux and Solaris have predefined PAM modules.

You can use one of these modules, or create one of your own. When creating your own modules, follow the guidelines in your operating system documentation on creating a PAM module.

> **i Note**
>
> PAM modules you create should comply with RFC 86.0 “Unified Login With Pluggable Authentication Modules (PAM).” SAP ASE supports the authentication management module of the RFC. It does not support the account management, session management, or password management modules.

6.5.1.1 Configuring Operating Systems

Configure your operating system to enable PAM support.

Context

- For Solaris, add the following line to `/etc/pam.conf`:

  ```
  ase auth required /user/lib/security/$ISA/pam_unix.so.1
  ```

- For Linux, create a new file called `/etc/pam.d/ase`, and add:

  ```
  auth required /lib/security/pam_unix.so
  ```

For more information on how to create these entries, see your operating system documentation.

6.5.1.2 Running a 32- and 64-bit Server on the Same Machine

$ISA is an environment variable that allows 32- and 64-bit libraries to run together.

Context

On Solaris 32-bit machines, $ISA is replaced by an empty string, while on 64-bit machines, it is replaced by the string “sparcv9”.

To use both 32- and 64-bit servers, place the 32-bit PAM module in a directory, and place the 64-bit version in a subdirectory of this directory.

The entry in `pam.conf` should look similar to:

```
$ ls /usr/lib/security/pam_sec.so.1
pam_sec.so.1 -> /SYBASE/pam_whatever_32bits.so.1
$ ls /usr/lib/security/sparcv9/pam_sec.so.1
pam_sec.so.1 -> /SYBASE/pam_sec_64bits.so.1
ase   auth   required /usr/lib/security/$ISA/pam_sec.so.1
```

> **i Note**
>
> $ISA is the only variable allowed in `pam.conf`.

6.5.1.3 Configure SAP ASE for PAM User Authentication

`enable pam user auth` enables PAM user authentication support.

The syntax is:

```
sp_configure "enable pam user auth", 0 | 1 | 2
```

where:

- 0 – disables PAM authentication. This is the default.
- 1 – indicates SAP ASE first attempts PAM authentication, and then uses `syslogins` authentication if PAM authentication fails.
- 2 – indicates only PAM authentication may be used.

> **i** Note
>
> When PAM is enabled, password management is delegated to the PAM service providers.

6.5.1.4 Logins and PAM User Accounts

After you have set `enable PAM user authentication` and completed the PAM configuration for both SAP ASE and the operating system, you must configure the user accounts.

The operating system or network security administrator creates and maintains user accounts in the PAM service provider, and the database administrator creates and maintains accounts in SAP ASE. Alternatively, the database administrator can choose administration options that allow flexibility with login accounts when integrating SAP ASE with external authentication mechanisms such as PAM. The database administrator continues to administer the account roles, default database, default language, and other login-specific attributes using traditional commands and procedures.

The following table describes updates to `syslogins` made at login time. It assumes that PAM user authentication is configured, the login is not restricted from using PAM, and you have not set the `create login` mapping

Does the row exist in syslogins?	PAM authentication succeeds?	Changes in syslogins
No	Yes	No change, login fails
No	No	No change, login fails
Yes	Yes	Update row if password has changed
Yes	No	No change

6.6 Enhanced Login Controls

Configure SAP ASE to allow the server-wide authentication mechanism according to the methods discussed in the LDAP and PAM sections earlier.

You can also configure SAP ASE to specify the authentication mechanism for each individual login on the server using enhanced login controls.

Login-specific controls may be useful when a server is transitioning between authentication mechanisms or for server-specific logins that local server administration may require; they are not associated with a centrally managed user login.

6.6.1 Forced Authentication

You can force a login to use a specific authentication process by using `alter login` and `create login`

Use these parameters for `alter login` and `create login`:

- `ASE` – use SAP ASE internal authentication using passwords from `syslogins` table.
- `LDAP` – use external authentication with an LDAP server.
- `PAM` – use external authentication with PAM.
- `ANY` – by default, users are authenticated using this authentication method. A user with ANY authentication means that SAP ASE checks if there is any external authentication mechanism defined, and if there is, it is used. Otherwise, it uses SAP ASE authentication.

SAP ASE checks for external authentication mechanisms in the following order:

1. LDAP.
2. Pluggable Authentication Modules (PAM). If both LDAP and PAM are enabled, PAM authentication is never attempted for a user.
3. If neither PAM nor LDAP is enabled, SAP ASE uses `syslogins` to authenticate the login.

Login accounts such as "sa" continue to be validated using the `syslogins` catalog. Only the SSO role can set authenticate for a login.

For example, the following authenticates the login with `alter login`:

```
alter login nightlyjob modify authenticate with ASE
sp_displaylogin "nightlyjob"
```

Displays output similar to:

```
Suid: 1234
Loginname: nightlyjob
Fullname: Batch Login
Default Database: master
. . .
Date of Last Password Change: Oct 2 2003 7:38 PM
Password expiration interval: 0
Password expired: N
Minimum password length:
Maximum failed logins: 0
Current failed login attempts:
```

```
Authenticate with: ASE
```

6.6.2 Map Logins Using sp_maplogin

Use `sp_maplogin` to map logins.

The syntax is:

```
sp_maplogin (authentication_mech | null),
(<client_username> | null), (<action> | <login_name> | null)
```

This example maps external user "jsmith" to the user "guest." Once authenticated, "jsmith" has the privileges of "guest." The audit login record shows both the `<client_username>` and the user name:

```
sp_maplogin NULL, "jsmith", "guest"
```

This example creates a new login for all external users authenticated with LDAP, if a login does not already exist:

```
sp_maplogin LDAP, NULL, "create login"
```

For more information, see *Reference Manual: Procedures*.

6.6.2.1 Display Mapping Information

`sp_helpmaplogin` displays mapping information.

The syntax is:

```
sp_helpmaplogin [ (authentication_mech | null), (client_username | null) ]
```

where:

`<client_username>` – is an external user name.

If you do not include any parameters, `sp_helpmaplogin` displays login information about all users currently logged in to SAP ASE. You can restrict the output to specific sets of client user names or authentication mechanisms by using the parameters listed above.

This displays information about all logins:

```
sp_helpmaplogin
```

```
authentication   client name   login name
--------------   -----------   ------------------
NULL             jsmith        guest
LDAP             NULL          create login
```

6.6.2.2 Determine the Authentication Mechanism

Use the `<@@authmech>` global variable to determine the authentication mechanism SAP ASE uses.

For example, if SAP ASE is enabled for LDAP user authentication with failover (`enable ldap user auth` = 2) and user "Joe" is an external user with authentication set to `ANY`, when Joe logs in, SAP ASE attempts to authenticate Joe, using LDAP user authentication. If Joe fails authentication as a user in LDAP, Joe is authenticated using SAP ASE authentication, and if that succeeds, he logs in successfully.

`<@@authmech>` global has this value:

```
select @@authmech
---------------------------------
ase
```

If SAP ASE is configured for strict LDAP user authentication (`enable ldap user auth` = 2) and Joe is added as a valid user in LDAP, when Joe logs in, the value for `@@authmech` is:

```
select @@authmech
---------------------------------
ldap
```

7 Manage User Permissions

Discretionary access controls (DACs) allow you to restrict access to objects and commands based on a user's identity, group membership and active roles.

The controls are "discretionary" because a user with a certain access permission, such as an object owner, can choose whether to pass that access permission on to other users.

The discretionary access control system recognizes the following types of users:

- Users possessing one or more system defined roles: system administrator, system security officer, operator, and other roles
- Database owners
- Database object owners
- Other users

System administrators (those users with sa_role) operate outside the DAC system and have access permissions on all database objects at all times except encryption keys (see *Database Encryption*). System security officers can always access the audit trail tables in the `sybsecurity` database to track accesses by system administrators.

If you have the `sa_role`, `all` grants permissions for `create database`, `set tracing`, and `connect` as well, if you issue the `grant` command in the `master` database.

Database owners do not automatically receive permissions on objects owned by other users; however, they can:

- Temporarily acquire all permissions of a user in the database by using the `setuser` command to assume the identity of that user.
- Permanently acquire permission on a specific object by using the `setuser` command to assume the identity of the object owner, and then using `grant` commands to grant the permissions.

Object owners can grant access to those objects to other users and can also grant other users the ability to pass the access permission to other users. You can give various permissions to users, groups, and roles with the `grant` command, and rescind them with the `revoke` command. Use `grant` and `revoke` to give users permission to:

- Create databases
- Create objects within a database
- Execute certain commands such as `dbcc` and `set proxy`
- Access specified tables, views, stored procedures, encryption keys, and columns

`grant` and `revoke` can also be used to set permissions on system tables.

For permissions that default to "public," no `grant` or `revoke` statements are needed.

Some commands can be used at any time by any user, with no permission required. Others can be used only by users of a particular status and they are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's role or status (as system administrator, database owner, system security officer, or database object

owner), and by whether the user was granted a role with permission that includes the option to grant that permission to other users.

You can also use views and stored procedures as security mechanisms.

Operations may differ when granular permissions is enabled.

Related Information

Views and Stored Procedures as Security Mechanisms [page 191]
Granular Permissions [page 241]

7.1 Granting Permissions for Creating Databases

Only a system administrator can grant permission to use the `create database` command.

Context

The user that receives `create database` permission must also be a valid user of the `master` database because all databases are created while using `master`. In many installations, the system administrator maintains a monopoly on `create database` permission to centralize control of database placement and database device space allocation. In these situations, a system administrator creates new databases on behalf of other users, and then transfers ownership to the appropriate user.

Procedure

1. Issue the `create database` command in the `master` database.
2. Switch to the new database with the `use` command.
3. Execute `sp_changedbowner`.

7.1.1 Change Database Ownership

Use `sp_changedbowner` to change the ownership of a database.

Often, system administrators create the user databases, then give ownership to another user after some of the initial work is complete. Only the system administrator can execute `sp_changedbowner`.

SAP suggests that you transfer ownership before the user has been added to the database, and before the user has begun creating objects in the database. The new owner must already have a login name on the server, but cannot be a user of the database, or have an alias in the database. You may have to use `sp_dropuser` or `sp_dropalias` before you can change a database's ownership, and you may have to drop objects before you can drop the user.

Issue `sp_changedbowner` in the database whose ownership is to be changed. The syntax is:

```
sp_changedbowner <loginame> [, true ]
```

This example makes "albert" the owner of the current database and drops aliases of users who could act as the old "dbo:"

```
sp_changedbowner albert
```

Include the `true` parameter to transfer aliases and their permissions to the new "dbo."

> **i Note**
>
> You cannot change the ownership of the `master`, `model`, `tempdb`, or `sybsystemprocs` databases and should not change the ownership of any other system databases.

7.2 Database Owner Privileges

The database owner has full privileges to do anything inside that database, and must explicitly grant permissions to other users with the `grant` command.

Database owners and system administrators are the only users who can grant object creation permissions to other users (except for `create encryption key` and `create trigger` permission which can only be granted by the system security officer).

Permission to use the following commands is automatically granted to the database owner and cannot be transferred to other users:

- `checkpoint`
- `dbcc`
- `alter database`
- `online database`
- `drop database`
- `dump database`
- `dump transaction`
- `grant` (object creation permissions)
- `load database`
- `load transaction`
- `revoke` (object creation permissions)
- `setuser`

Database owners can grant or revoke permission to:

- Use these commands: `create default`, `create procedure`, `create rule`, `create table`, `create view`.
  Database owners can grant permission to use `create database`, `set tracing`, and `connect` if they have the `sa_role` and are in the `master` database.
- `all` – if you are the database owner, `all` grants permissions for all `create` commands except `create database`, `create trigger` and `create encryption key`.
- default permissions on system tables
- Use `dbcc` commands:`checkalloc`, `checkcatalog`, `checkdb`, `checkindex`, `checkstorage`, `checktable`, `checkverify`, `fix_text`, `indexalloc`, `reindex`, `tablealloc`, `textalloc`, `tune`

Permissions are granted to or revoked from other database users by object owners, database owners, users who were granted permissions with `grant` option, the system administrator, or a system security officer. These users are specified by user name, group name, or the keyword `public`. All users inherit the permissions granted to the roles assigned to them after they have activated those roles.

7.3 Database Object Owner Privileges

A user who creates a database object (a table, view, encryption key, or stored procedure) owns the object and is automatically granted all object access permissions on it.

Users other than the object owner, including the owner of the database, are automatically denied all permissions on that object, unless they are explicitly granted by either the owner or a user who has `grant` permission on that object.

As an example, suppose that Mary is the owner of the `pubs2` database, and has granted Joe permission to create tables in it. Now Joe creates the table `new_authors`; he is the owner of this database object.

Initially, object access permissions on `new_authors` belong only to Joe. Joe can grant or revoke object access permissions for this table to other users.

The following object altering permissions default to the owner of a table and cannot be transferred to other users:

- `alter table`
- `drop table`
- `create index`

Permission to use the `grant` and `revoke` commands to grant specific users `select`, `insert`, `update`, `delete`, `references`, `decrypt`, `truncate table`, `update statistics`, `delete statistics`, and `execute` permissions on specific database objects can be transferred, using the `grant with grant option` command.

Permission to `drop` an object—a table, view, index, stored procedure, rule, encryption key, trigger, or default—defaults to the object owner and cannot be transferred.

7.4 Permissions on System Procedures

Set permissions on system procedures in the `sybsystemprocs` database, where the system procedures are stored.

Security-related system procedures can be run only by system security officers. Certain other system procedures can be run only by system administrators.

Some of the system procedures can be run only by database owners. These procedures make sure that the user executing the procedure is the owner of the database from which they are being executed.

Other system procedures can be executed by any user who has been granted permission. A user must have permission to execute a system procedure in all databases, or in none of them.

Users who are not listed in `sybsystemprocs..sysusers` are treated as "guest" in `sybsystemprocs`, and are automatically granted permission on many of the system procedures. To deny a user permission on a system procedure, the system administrator must add him or her to `sybsystemprocs..sysusers` and issue a `revoke` statement that applies to that procedure. The owner of a user database cannot directly control permissions on the system procedures from within his or her own database.

7.5 Grant or Revoke Permissions

Use the `grant` and `revoke` commands to control permissions.

These permissions are controlled with the `grant` and `revoke` commands:

- Object access permissions
- Permission to select from functions
- Permission to execute commands
- Permission to execute `dbcc` commands
- Permission to execute some `set` commands
- Default permissions on system tables

Each database has its own independent protection system. Having permission to use a certain command in one database does not give you permission to use that command in other databases.

7.5.1 Object Access Permissions

Object access permissions regulate the use of certain commands that access certain database objects.

For example, you must explicitly be granted permission to use the `select` command on the `authors` table. Object access permissions are granted and revoked by the object owner (and system administrators or system security officers), who can grant them to other users.

Table 13: Permissions and the objects to which they apply

Permission	Object
`select`	Table, view, column
`update`	Table, view, column
`insert`	Table, view
`delete`	Table, view
`references`	Table, column
`execute`	Stored procedure
`truncate table`	Table
`delete statistics`	Table
`update statistics`	Table
`decrypt`	Table, view, column
`select`	Encryption key

The `references` permission refers to referential integrity constraints that you can specify in an `alter table` or `create table` command. The `decrypt` permission refers to the permission required to decrypt an encrypted column. An encryption key's `select` permission refers to the permissions required to use encryption keys in `create table`, `alter table` or `select into` command to encrypt columns. The other permissions refer to SQL commands. Object access permissions default to the object's owner, or system administrators or system security officers for `decrypt` on an encrypted column and `select` on an encryption key, and can be granted to other users.

If several users grant access to an object to a particular user, the user's access remains until access is revoked by all those who granted access. If a system administrator revokes access, the user is denied access, even though other users have granted access.

Use the `grant` command to grant object access permissions. See the *Reference Manual: Commands*.

You can grant `select`, `update` and `delete` permission using a `where` clause that can restrict access on a row by row basis based on the condition in the `where` clause.

7.5.1.1 Concrete Identification

When the user creates an object, the server associates both the owner's database user ID (`<uid>`) and the creator's login name with the object in the `sysobjects` table. This information concretely identifies the object as belonging to that user.

Concrete identification allows the server to recognize when permissions on an object can be granted implicitly.

> **i Note**
>
> SAP ASE identifies users during a session by login name. This identification applies to all databases in the server.

If a user creates a table and then creates a procedure that accesses the table, any user who is granted permission to execute the procedure does not need permission to access the object directly. For example, by giving user "mary" permission on `proc1`, she can see the `id` and `descr` columns from `table1`, though she does not have explicit select permission on the table:

```
create table table1 (id     int,
                     amount money,
                     descr    varchar(100))
```

```
create procedure proc1 as select id, descr from table1
```

```
grant execute on proc1 to mary
```

There are, however, some cases where implicit permissions are only useful if the objects can be concretely identified. One case is where aliases and cross-database object access are both involved.

7.5.1.2 Special Requirements for SQL92 Standard Compliance

When you use the `set` command to turn `ansi_permissions` on, additional permissions are required for `update` and `delete` statements.

These are the ANSI permissions for update and delete:

	Permissions required: *set ansi_permissions off*	**Permissions required:** *set ansi_permissions on*
`update`	`update` permission on columns where values are being set	`update` permission on columns where values are being set and `select` permission on all columns appearing in the `where` clause `select` permission on all columns on the right side of the `set` clause
`delete`	`delete` permission on the table	`delete` permission on the table from which rows are being deleted and `select` permission on all columns appearing in the `where` clause

If `ansi_permissions` is on and you attempt to update or delete without having all the additional `select` permissions, the transaction is rolled back and you receive an error message. If this occurs, the object owner must grant you `select` permission on all relevant columns.

7.5.1.3 Grant Object Access Permissions Examples

Examples of granting object access permissions.

This statement gives Mary and the "sales" group permission to insert into and delete from the `titles` table:

```
grant insert, delete
on titles
to mary, sales
```

This statement gives Harold permission to use the stored procedure `makelist`:

```
grant execute
on makelist
to harold
```

This statement grants permission to execute the custom stored procedure `sa_only_proc` to users who have been granted the system administrator role:

```
grant execute
on sa_only_proc
to sa_role
```

This statement gives Aubrey permission to select, update, and delete from the `authors` table and to grant the same permissions to other users:

```
grant select, update, delete
on authors
to aubrey
with grant option
```

This statement grants permission to the payroll employees to update salaries during December:

```
grant update (salary)
on employee
where date_part(month, getdate()) = 12
to payroll_role
```

7.5.1.4 Revoke Object Access Permissions Examples

Examples of Revoking Object Access Permissions.

These two statements both revoke permission for all users except the table owner to update the `price` and `total_sales` columns of the `titles` table:

```
revoke update
on titles (price, total_sales)
from public
```

This statement revokes permission from Clare to update the `authors` table, and simultaneously revokes that permission from all users to whom she had granted that permission:

```
revoke update
on authors
```

```
from clare
cascade
```

This statement revokes permission from operators to execute the custom stored procedure `new_sproc`:

```
revoke execute
on new_sproc
from oper_role
```

7.5.2 Grant Permissions on dbcc Commands

System administrators can grant the permission to execute `dbcc` commands to users and roles that do not have system administrator-level privileges in Adaptive Server.

This discretionary access control allows system administrators to control access to database objects or to certain database- and server-level actions.

See the *Reference Manual: Commands* for the complete `dbcc` syntax.

7.5.2.1 Server-wide and Database-Specific dbcc Commands

`dbcc` commands are either database specific or server-wide.

- Database-specific `dbcc` commands are execute on a particular target database (for example, `checkalloc`, `checktable`, `checkindex`, `checkstorage`, `checkdb`, `checkcatalog`, `checkverify`, `fix_text`, `indexalloc`, `reindex`, `tablealloc`, and `textalloc`). Although these commands are database-specific, only system administrators can grant or revoke them.
- Server-wide `dbcc` commands such as `tune` are effective server-wide and are not associated with any particular database. These commands are granted server-wide by default and are not associated with any database.

System administrators can allow users to execute the `dbcc` command in all databases by making them valid users in those databases. However, it may be more convenient to `grant dbcc` to roles instead of individual users, since this allows users to use databases as a "guest" user instead of requiring that they each be added manually to the database.

Since roles are automatically added as users in a database on their first `grant` in a database, there are no additional requirements when roles are granted `dbcc` privileges. Logins must be valid users in the database where permissions are granted. Valid users include "guest."

For server-wide `dbcc` commands, the login must be a valid user in `master`, and the system administrator must be in `master` when granting the permission.

For database-specific `dbcc` commands the login should be a valid user in the target database.

From a security administration perspective, system administrators may prefer to grant permission to execute database-specific `dbcc` commands server-wide. For example, you can execute `grant dbcc checkstorage`

on all databases to a user-defined role called `storage_admin_role`, thereby eliminating the need to execute `grant dbcc checkstorage` to `storage_admin_role` in every database.

The following commands are effective server-wide, but are not database-specific:

- Server-wide `dbcc` commands such as `tune`.
- Database-specific `dbcc` commands that are granted server-wide, such as `grant dbcc checkstorage` granted to `storage_admin_role`.

7.5.3 Permissions on System Tables

Permissions for use of the system tables can be controlled by the database owner, just like permissions on any other tables.

When a database is created, `select` permission on some system tables is granted to `public`, and `select` permission on some system tables is restricted to administrators. For some other tables, a few columns have restricted `select` permissions for `public`.

To determine the current permissions for a particular system table, execute:

```
sp_helprotect <system_table_name>
```

For example, to check the permissions of `syssrvroles` in the master database, execute:

```
use master
go
sp_helprotect syssrvroles
go
```

By default, no user—including the database owner—can modify the system tables directly. Instead, the T-SQL commands and the system procedures supplied with SAP ASE modify the system tables. This helps guarantee integrity.

> ⚠ Caution
>
> Although SAP ASE provides a mechanism that allows you to modify system tables, SAP strongly recommends that you do not do so.

7.5.3.1 Grant Default Permissions to System Tables and Stored Procedures

The `grant` and `revoke` commands include the `default permissions` parameter.

`installmodel` or `installmaster` do not grant default permissions on any system tables (see the table below). Instead, the default permissions on the system tables are assigned when SAP ASE builds a new database. The partial syntax is:

```
grant default permissions on system tables
```

```
revoke default permissions on system tables
```

where `default permissions on system tables` specifies that you grant or revoke the default permissions for the following system tables when you issue it from any database:

sysalternates	sysjars	sysquerymatrics	systhresholds
sysattributes	syskeys	sysqueryplans	systypes
syscolumns	syslogs	sysreferences	sysusermessages
syscomments	sysobjects	sysroles	sysusers
sysconstraints	syspartitionkeys	syssegments	sysxtypes
sysdepends	syspartitions	sysslices	
sysgams	sysprocedures	sysstatistics	
sysindexes	sysprotects	systabstats	

Default permissions applies `select` to public on all system tables, with these exceptions:

- Revokes `select` on `syscolumns(encrkeyid)` from public
- Revokes `select on syscolumns(encrkeydb)` from public
- Grants `select on syscolumns` to `sso_role`
- Revokes `sysobjects(audflags)` permissions from public
- Grants permissions for `sysobjects` to `sso_role`
- Revokes `select on` all columns of `sysencryptkeys` from public
- Grants `select on` all columns of `sysencryptkeys` to sso_role

If you run this command from the `master` database, default permissions for the following system tables are granted or revoked:

syscharsets	syslanguages	sysmessages	sysservers
sysconfigures	syslisteners	sysmonitor	syssessions
syscurconfigs	syslocks	sysprocesses	syssrvroles
sysdatabases	syslogin	sysremotelogins	systimeranges
sysdevices	sysloginrole	sysresourcelimits	systransactions
sysengines	syslogshold	syssecmechs	sysusages

The command also makes the following changes:

- Revokes `select` on `sysdatabases(audflags)` from public
- Revokes `select` on `syscolumns(encrkeyid)` from public
- Revokes `select` on `syscolumns(encrkeydb)` from public
- Grants `select` on `syscolumns` to `sso_role`
- Revokes `select` on `sysdatabases(deftabaud)` from public
- Revokes `select` on `sysdatabases(defvwaud)` from public
- Revokes `select` on `sysdatabases(defpraud)` from public
- Revokes `select` on `sysdatabases(audflags2)` from public

- Grants `select` on `sysdatabases` to `sso_role`.
- Revokes `select` on `syslogins(password)` to public
- Revokes `select` on `syslogins(audflags)` from public
- Grants `select` on `syslogins` to sso_role
- Revokes `select` on `syslisteners(net_type)` from public
- Revokes `select` on `syslisteners(address_info)` from public
- grant `select` on `syslisteners` to sso_role
- Revokes `select` on `syssrvroles(srid)` from public
- Revokes `select` on `syssrvroles(name)` from public
- Revokes `select` on `syssrvroles(password)` from public
- Revokes `select` on `syssrvroles(pwdate)` from public
- Revokes `select` on `syssrvroles(status)` from public
- Revokes `select` on `syssrvroles(logincount)` from public
- grant `select` on `syssrvroles` to sso_role
- Revokes `select` on `sysloginroles(suid)` from public
- Revokes `select` on `sysloginroles(srid)` from public
- Revokes `select` on `sysloginroles(status)` from public
- Revokes `select` on `sysloginroles` to sso_role

7.5.4 Combine grant and revoke Statements

Assign specific permissions to specific users, or, if most users are going to be granted most privileges, it may be easier to assign all permissions to all users, and then revoke specific permissions from specific users.

For example, a database owner can grant all permissions on the `titles` table to all users by issuing:

```
grant all
on titles
to public
```

The database owner can then issue a series of `revoke` statements, for example:

```
revoke update
on titles (price, advance)
from public
```

```
revoke delete
on titles
from mary, sales, john
```

`grant` and `revoke` statements are order-sensitive: in case of a conflict, the most recently issued statement supersedes all others.

> **i** Note
>
> Under SQL rules, you must use the `grant` command before using the `revoke` command, but the two commands cannot be used within the same transaction. Therefore, when you grant "public" access to objects, and then revoke that access from an individual, there is a short period of time during which the

individual has access to the objects in question. To prevent this situation, use the `create schema` command to include the `grant` and `revoke` clauses within one transaction.

7.5.5 Permission Order and Hierarchy

`grant` and `revoke` statements are sensitive to the order in which they are issued.

For example, if Jose's group has been granted `select` permission on the `titles` table and then Jose's permission to `select` the `advance` column has been revoked, Jose can `select` all the columns except `advance`, while the other users in his group can still `select` all the columns.

A `grant` or `revoke` statement that applies to a group or role changes any conflicting permissions that have been assigned to any member of that group or role. For example, if the owner of the `titles` table has granted different permissions to various members of the `sales` group, and wants to standardize, he or she might issue the following statements:

```
revoke all on titles from sales
grant select on titles(title, title_id, type,
        pub_id)
    to sales
```

Similarly, a `grant` or `revoke` statement issued to `public` changes, for all users, all previously issued permissions that conflict with the new regime.

The same `grant` and `revoke` statements issued in different orders can create entirely different situations. For example, the following set of statements leaves Jose, who belongs to the `public` group, without any `select` permission on `titles`:

```
grant select on titles(title_id, title) to jose
revoke select on titles from public
```

In contrast, the same statements issued in the opposite order result in only Jose having `select` permission and only on the `title_id` and `title` columns:

```
revoke select on titles from public
grant select on titles(title_id, title) to jose
```

When you use the keyword `public` with `grant`, you are including yourself. With `revoke` on object creation permissions, you are included in `public` unless you are the database owner. With `revoke` on object access permissions, you are included in `public` unless you are the object owner. You may want to deny yourself permission to use your own table, while giving yourself permission to access a view built on it. To do this, you must issue `grant` and `revoke` statements explicitly setting your permissions. You can reinstitute the permission with a `grant` statement.

> **i Note**
>
> `grant dbcc` and `set proxy` issue the following warning when they are issued while `set fipsflagger` option is enabled:
>
> ```
> SQL statement on line number 1 contains Non-ANSI text. The error is caused
> due to the use of DBCC.
> ```

7.6 Acquire the Permissions of Another User

Two ways are provided to acquire another user's identity and permissions status.

- A database owner can use the `setuser` command to "impersonate" another user's identity and permissions status in the current database.
- proxy authorization allows one user to assume the identity of another user on a server-wide basis.

7.6.1 setuser Command

The Database Owner uses `setuser` to adopt the identity of another user.

Use `setuser` to:

- Access an object owned by another user
- Grant permissions on an object owned by another user
- Create an object that will be owned by another user
- Temporarily assume the DAC permissions of another user for some other reason

While the `setuser` command enables the database owner to automatically acquire another user's DAC permissions, the command does not affect the roles that have been granted.

`setuser` permission defaults to the database owner and cannot be transferred. The user being impersonated must be an authorized user of the database. SAP ASE checks the permissions of the user being impersonated.

System administrators can use `setuser` to create objects that will be owned by another user. However, system administrators operate outside the DAC permissions system; therefore, they need not use `setuser` to acquire another user's permissions. The `setuser` command remains in effect until another `setuser` command is given, the current database is changed, or the user logs off.

> **i** Note
>
> `setuser` requires that you disable quoted identifiers with `set quoted_identifier off`, and that the target user against whom you are issuing `setuser` already exists in the database.

The syntax is:

```
setuser ["<user_name>"]
```

where `<user_name>` is a valid user in the database that is to be impersonated.

To reestablish your original identity, use `setuser` with no value for `<user_name>`.

This example shows how the database owner would grant Joe permission to read the `authors` table, which is owned by Mary:

```
setuser "mary"
grant select on authors to joe
setuser     /*reestablishes original identity*/
```

7.6.2 Proxy Authorization

With the proxy authorization capability of SAP ASE, system security officers can grant selected logins the ability to assume the security context of another user, and an application can perform tasks in a controlled manner on behalf of different users.

If a login has permission to use proxy authorization, the login can impersonate any other login in SAP ASE.

> ⚠ Caution
>
> The ability to assume another user's identity is extremely powerful and should be limited to trusted administrators and applications. `grant set proxy ... restrict role` can be used to restrict which roles users cannot acquire when switching identities.

A user executing `set proxy` or `set session authorization` operates with both the login name and server user ID of the user being impersonated. The login name is stored in the `name` column of `master..syslogins` and the server user ID is stored in the `suid` column of `master..syslogins`. These values are active across the entire server in all databases.

> i Note
>
> `set proxy` and `set session authorization` are identical in function and can be used interchangeably. The only difference between them is that `set session authorization` is ANSI-SQL92-compatible, and `set proxy` is a Transact-SQL extension.

7.6.2.1 Use set proxy to Restrict Roles

Grant `set proxy...restrict role` to restrict which roles cannot be acquired when switching identities.

The syntax for `set proxy` is:

```
grant set proxy to <user> | <role>
    [restrict role <role_list> | all | system]
```

where:

- `<role_list>` – list of roles you are restricting for the target login. The grantee must have all roles on this list, or the `set proxy` command fails.
- `<all>` – ensures the grantee can run `set proxy` only for those users who have the same roles, or a subset of the roles, as the grantee.
- `<system>` – ensures the grantee has the same set of system roles as the target login.

For example, this grants `set proxy` to user "joe" but restricts him from switching identities to any user with the `sa`, `sso`, or `admin` roles (however, if he already has these roles, he can `set proxy` for any user with these roles):

```
grant set proxy to joe
restrict role sa_role, sso_role, admin_role
```

When “joe” tries to switch his identity to a user with admin_role (in this example, `Our_admin_role`), the command fails unless he already has `admin_role`:

```
set proxy Our_admin_role
```

```
Msg 10368, Level 14, State 1:
Server 's', Line 2:Set session authorization permission denied because the
target login has a role that you do not have and you have been restricted from
using.
```

After “joe” is granted the `admin_role` and retries the command, it succeeds:

```
grant role admin_role to joe
set proxy Our_admin_role
```

For more information about the set proxy command, see the *Reference Manual: Commands*.

7.6.2.2 Execute Proxy Authorization

Follow these rules when executing `set proxy` or `set session authorization`.

- You cannot execute `set proxy` or `set session authorization` from within a transaction.
- You cannot use a locked login for the proxy of another user. For example, if “joseph” is a locked login, the following command is not allowed:

  ```
  set proxy "joseph"
  ```
- You can execute `set proxy` or `set session authorization` from any database you are allowed to use. However, the `<login_name>` you specify must be a valid user in the database, or the database must have a “guest” user defined for it.
- Only one level is permitted; to impersonate more than one user, you must return to your original identity between impersonations.
- If you execute `set proxy` or `set session authorization` from within a procedure, your original identity is automatically resumed when you exit the procedure.

If you have a login that has been granted permission to use `set proxy` or `set session authorization`, you can set proxy to impersonate another user. The following is the syntax, where `<login_name>` is the name of a valid login in `master..syslogins`:

```
set proxy <login_name>
```

or

```
set session authorization <login_name>
```

Enclose the login name in quotation marks.

For example, to set proxy to “mary,” execute:

```
set proxy "mary"
```

After setting proxy, check your login name in the server and your user name in the database. For example, assume that your login is “ralph” and that you have been granted `set proxy` authorization. You want to

execute some commands as “sallyn” and as “rudolph” in `pubs2` database. “sallyn” has a valid name (“sally”) in the database, but Ralph and Rudolph do not. However, `pubs2` has a “guest” user defined. You can execute:

```
set proxy "sallyn"
go
use pubs2
go
select suser_name(), user_name()
go
```

```
------------------------------ ------------------
sallyn                         sally
```

To change to Rudolph, you must first change back to your own identity. To do so, execute:

```
set proxy "ralph"
select suser_name(), user_name()
go
```

```
------------------------------ ------------------
ralph                          guest
```

Notice that Ralph is a “guest” in the database.

Then execute:

```
set proxy "rudolph"
go
select suser_name(), user_name()
go
```

```
------------------------------ ------------------
rudolph                        guest
```

Rudolph is also a guest in the database because Rudolph is not a valid user in the database.

Now, impersonate the “sa” account. Execute:

```
set proxy "ralph"
go
set proxy "sa"
go
select suser_name(), user_name()
go
```

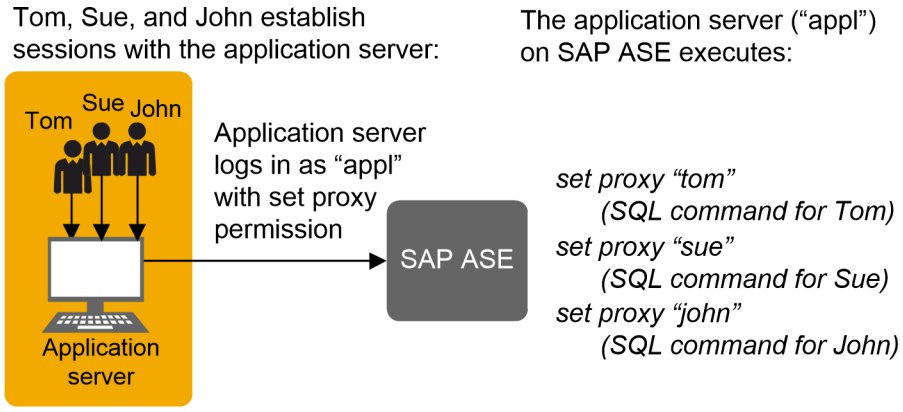
```
-------------------------- ------------------
sa                         dbo
```

7.6.2.3 Proxy Authorization for Applications

Using proxy authorization, an application server can log in to an SAP ASE server with a generic login to execute procedures and commands for users.

The diagram below shows an application server logging in to SAP ASE server with the generic login “appl” to execute procedures and commands for several users. While “appl” impersonates Tom, the application has

Tom's permissions. Likewise, when "appl" impersonates Sue and John, the application has only Sue's and John's permissions, respectively.



7.6.3 Resolving Qualified Object Names on Behalf of Another User

SAP ASE allows a database owner to specify object name resolution for a specified user so that SAP ASE resolves qualified object names on behalf of this user.

Context

That is, a database owner can specify object name resolution for a user so that SAP ASE resolves qualified object names on behalf of another user, allowing the first user to execute the command even though they have read-only permission on the object.

Use `sp_modifyuser` to map one user to resolve qualified objects as another user. The syntax is:

```
sp_modifyuser '<username_1>', 'resolve as', '<username_2>'
```

> **i** Note
>
> You must enable the `enable functionality group` before issuing `sp_modifyuser ... resolve as`.

Procedure

For example, if you issue this for user_A_RO:

```
sp_modifyuser 'user_A_RO', 'resolve as', 'user_B'
```

When user user_A_RO issues this command:

```
select cname from customer c, orders o
where c.cid = o.oid and o.date = 'today'
```

SAP ASE processes the command as if user_A_RO issued:

```
select cname from user_B.customer c, user_B.orders o
where c.cid = o.oid and o.date = 'today'
```

7.7 Changing Database Object Ownership

A system security officer or database owner can transfer the ownership of database objects using the `alter... modify owner` command.

The command lets a database administrator manage the assignment of objects due to employee changes or to separate the creation ownership of database objects. For example, a key custodian can create an encryption key and then transfer the ownership of the encryption key to another user.

7.7.1 Supported Object Types

The ownership of objects can be transferred from one owner to another.

Objects for which the ownership can be changed explicitly:

- User tables
- Proxy tables
- Views
- Stored procedures
- User-defined functions
- Defaults
- Rules
- User-defined datatypes
- Encryption keys

> **i** Note
>
> The ownership of objects not listed below cannot be changed.

Dependent objects for which the ownership cannot be changed explicitly. These objects are transferred implicitly when the ownership is the same as the explicitly transferred object:

- Triggers
  The ownership of a dbo-owned trigger cannot be altered if the trigger was created for a non-dbo-owned table/view.
- Declarative objects that are defined during the table/view creation

- Defaults
- Decrypt_defaults
- Check constraints
- Reference constraints
- Partition conditions
- Computed columns

7.7.2 Authorization to Transfer Ownership

System security officers have authorization to transfer ownership of all objects for which ownership transfer is supported. Database owners have authorization to transfer ownership of objects, other than encryption keys.

- Database owners can transfer ownership with these restrictions:
  - The database object owner cannot transfer the ownership of objects concretely owned by the database owner.
    An object is identified as concretely owned by a database owner if it carries the database owner user ID as `sysobjects.uid`, and null or the database owner's user name as `sysobjects.loginame`.
  - A user aliased to the database owner cannot transfer the ownership of objects created by the database owner or concretely owned by the user.
    Database owner-created objects have a null value in `sysobjects.loginame`. Objects concretely owned by a user carries the user's username in `sysobjects.loginame`.

Use `sp_helpuser` to search for and list objects and corresponding owners.

7.7.3 Transfer Ownership

Ownership transfer can be specific to an individual object, or multiple objects can be transferred in one command. Use `preserve permissions` to preserve explicitly granted permissions of an object.

For syntax, see `alter...modify owner` in *Reference Manual: Commands*.

In this example, the database owner transfers a table owned by john to eric.

```
alter table john.table_audit modify owner eric
```

To transfer the ownership of all tables owned by john to eric, a system security officer can execute:

```
alter table john.* modify owner eric
```

To transfer the ownership of all objects owned by john to eric, a system security officer can execute:

```
alter all john.* modify owner eric
```

7.7.3.1 Transfer Ownership of Objects in the System Database

Change the ownership of objects in system databases supplied by SAP.

Use caution to change the ownership of objects in the following system databases that are supplied and managed by SAP:

- `sybsecurity`
- `sybsystemdb`
- `model`
- `sybsystemprocs`
- `sybsyntax`
- `dbccdb`
- `tempdb`

Do not change the ownership of these system objects that are supplied and managed by SAP:

- User tables with `spt_` prefix
- System stored procedures with `sp_` prefix

Changing the ownership of these objects can make the system unusable.

7.7.3.2 Transfer Ownership of Database Owner Objects

The database owner of nonsystem objects can transfer ownership using the parameter `dbo.object_name`. You cannot transfer the ownership of multiple objects using `*`.

7.7.3.3 preserve permissions Command

Specify `preserve permissions` to preserve all explicitly granted or revoked permissions on an object.

For example, bill granted `select` permission of table `bill_table` to mark. mark then granted `select` permission on table `bill_table` to john. If the ownership of the table is then transferred to eric with `preserve permissions` specified, mark and john retain their permission of `bill_table`.

In the following example, the system security officer transfers the ownership of view `bill.vw_author` to eric while keeping all existing explicitly granted permissions.:

```
alter view bill.vw_author_in_ca modify owner eric
    preserve permissions
```

Implicit permissions are not preserved when `preserve permissions` is specified.

For example, bill owns table `bill.encr_table` which has encrypted columns and the `restricted decrypt permission` configure option is set to 1. If the system security officer explicitly granted decrypt permission on `bill.encr_table` to bill, bill has the permissions alter, delete, insert, references, select, and update which he

accrued through his ownership. He also has decrypt permission which he accrued through explicit granting by the system security officer. After the system security officer transfers the ownership on `bill.encr_table` to eric with `preserve permissions`, bill loses all permissions on the table except the decrypt permission.

When `preserve permissions` is not specified, after the ownership transfer, the previous owner loses permissions on the object, that are implicitly accrued through ownership. The new implicitly accrues permissions by being given ownership of the object.

> **i** Note
>
> For permissions that cannot be accrued through ownership, such as decrypt permissions, the system security officer or database owner must explicit again grant permission of the objects to the new owner.

7.7.3.4 Security Issues

The system security officer or database owner should be aware of possible security issues.

For example, alice is a user in the Accounting database and has no access to the payroll data. She could create the procedure alicep that selects name and salary from `Accounting.dbo.payroll`, and then grant execute on alicep to public. If the system security officers accidentally changes the ownership of alicep to bill, a privileged user with access to the payroll data with preserve permissions option, all users can access the payroll information by executing the malicious procedure alicep because all the permissions are set to be preserved after the ownership change.

To avoid unauthorized usage, the system security officers or database owner can check existing permissions on an object using `sp_helprotect`.

7.7.3.5 Transfer Ownership of Encryption Keys

System security officers and key owners can use `alter encryption key` or `alter... modify owner` to transfer encryption keys.

For information about the alter encryption key command, see *Reference Manual: Commands*.

7.7.3.5.1 Encryption Key Copy Owners

When using the `alter... modify owner` command, the user that has been assigned a key copy cannot be the new owner of the encryption key.

After the owner of a encryption key changes, the assignees of key copies do not change. For example, user bill owns an encryption key named `bill.encrkey` and creates one key copy of the key, which he assigns to mark. After bill transfers the ownership of `bill.encrkey` to eric, mark still owns a copy of `bill.encrkey`.

7.8 Report on Permissions

Use system procedures and system tables to report information about proxies, object creation, and object access permissions.

To report information on	Use
Proxies	system tables
Users and processes	`sp_who`
Permissions on database objects or users	`sp_helprotect`
Permissions on specific tables	`sp_helprotect`

7.8.1 Query the sysprotects Table for Proxy Authorization

To display information about permissions that have been granted to—or revoked from—users, groups, and roles, query the `sysprotects` table.

The `action` column specifies the permission. For example, the `action` value for `set proxy` or `set session authorization` is equal to 167.

When executing this query:

```
select * from sysprotects where action = 167
```

The results provide the user ID of the user who granted or revoked the permission (column `grantor`), the user ID of the user who has the permission (column `uid`), and the type of protection (column `protecttype`). The `protecttype` column can contain these values:

- 0 for `grant with grant`
- 1 for `grant`
- 2 for `revoke`

For more information about the `sysprotects` table, see the *Reference Manual: Building Blocks*.

7.8.2 Display Information About Users and Processes

`sp_who` displays information about all current SAP ASE users and processes or about a particular user or process.

The results of `sp_who` include the `loginame` and `orignamе`. If a user is operating under a proxy, `origname` contains the name of the original login. For example, assume that "ralph" executes the following, then executes some SQL commands:

```
set proxy susie
```

`sp_who` returns "susie" for `loginame` and "ralph" for `origname`.

`sp_who` queries the `master..sysprocesses` system table, which contains columns for the server user ID (`suid`) and the original server user ID (`origsuid`).

For more information, see `sp_who` in the *Reference Manual: Procedures*.

7.8.3 Report Permissions on Database Objects or Users

Use `sp_helprotect` to report on permissions by database object or by user, and (optionally) by user for a specified object. Any user can execute this procedure.

For example, suppose you issue the following series of `grant` and `revoke` statements:

```
grant select on titles to judy
grant update on titles to judy
revoke update on titles(contract) from judy
grant select on publishers to judy
    with grant option
```

To determine the permissions Judy now has on each column in the `titles` table, enter:

```
sp_helprotect titles, judy
```

```
grantor grantee type   action   object     column    predicate   grantable
------- ------ -----   ------   ------     ------    -------    ----------
dbo     judy   Grant   Select   titles     All         NULL       FALSE
dbo     judy   Grant   Update   titles     advance     NULL       FALSE
dbo     judy   Grant   Update   titles     notes       NULL       FALSE
dbo     judy   Grant   Update   titles     price       NULL       FALSE
dbo     judy   Grant   Update   titles     pub_id      NULL       FALSE
dbo     judy   Grant   Update   titles     pubdate     NULL       FALSE
dbo     judy   Grant   Update   titles     title       NULL       FALSE
dbo     judy   Grant   Update   titles     title_id    NULL       FALSE
dbo     judy   Grant   Update   titles     total_sales NULL       FALSE
dbo     judy   Grant   Update   titles     type        NULL       FALSE
```

The first row shows that the database owner ("dbo") gave Judy permission to select all columns of the `titles` table. The rest of the lines indicate that she can update only the columns listed in the display. Judy cannot give `select` or `update` permissions to any other user.

To see Judy's permissions on the `publishers` table, enter:

```
sp_helprotect publishers, judy
```

In this display, the `grantable` column indicates TRUE, meaning that Judy can grant the permission to other users.

```
grantor grantee type   action   object     column   predicate    grantable
------- ------ -----   ------   ------     ------   -------   -------
dbo     judy    Grant   Select  publishers   all       NULL        TRUE
```

See *Reference Manual: Procedures* for syntax.

7.8.4 Report Permissions on Specific Tables

Use `sp_helprotect` to return permissions information about a specified table.

To see permissions on the `sales` table, enter:

```
sp_tables sales
```

```
grantor grantee type       action        object  column  predicate  grantable----
--- ------ -----   ------   --------      ------   -------     -------
dbo     guest  Grant    Delete            sales   All     NULL       FALSE
dbo     guest  Grant    Delete Statistics sales   All     NULL       FALSE
dbo     guest  Grant    Insert            sales   All     NULL       FALSE
dbo     guest  Grant    References        sales   All     NULL       FALSE
dbo     guest  Grant    Transfer Table    sales   All     NULL       FALSE
dbo     guest  Grant    Transfer Table    sales   All     NULL       FALSE
dbo     guest  Grant    Update            sales   All     NULL       FALSE
dbo     guest  Grant    Update Statistics sales   All     NULL       FALSE
dbo     public Grant    Select            sales   All     NULL       FALSE
(1 row affected)
(return status = 0)
```

For more information about the output of `sp_helprotect`, see the *Reference Manual: Procedures*.

7.9 Views and Stored Procedures as Security Mechanisms

Views and stored procedures can serve as security mechanisms. You can give users controlled access to database objects via a view or stored procedure without granting them direct access to the data.

For example, you might give a clerk `execute` permission on a procedure that updates cost information in a `projects` table without letting the user see confidential data in the table. To use this feature, you must own the procedure or view as well as its underlying objects. If you do not own the underlying objects, users must have permission to access the objects. For more information about when permissions are required, see *Understanding Ownership Chains*.

SAP ASE makes permission checks, as required, when the view or procedure is used. When you create the view or procedure, SAP ASE makes no permission checks on the underlying objects.

Related Information

Understanding Ownership Chains [page 194]

7.9.1 Using Views as Security Mechanisms

Through a view, users can query and modify only the data they can see. The rest of the database is neither visible nor accessible.

Permission to access the view must be explicitly granted or revoked, regardless of the permissions on the view's underlying tables. If the view and underlying tables are owned by the same owner, no permissions need to be given on the underlying tables. Data in an underlying table that is not included in the view is hidden from users who are authorized to access the view but not the underlying table.

By defining different views and selectively granting permissions on them, a user (or any combination of users) can be restricted to different subsets of data. Access can be restricted to:

- A subset of the rows of a base table (a value-dependent subset). For example, you might define a view that contains only the rows for business and psychology books to keep information about other types of books hidden from some users.
- A subset of the columns of a base table (a value-independent subset). For example, you might define a view that contains all the rows of the `titles` table, but omits the `price` and `advance` columns, since this information is sensitive.
- A row-and-column subset of a base table.
- The rows that qualify for a join of more than one base table. For example, you might define a view that joins the `titles`, `authors`, and `titleauthor` tables. This view hides personal data about authors and financial information about the books.
- A statistical summary of data in a base table. For example, you might define a view that contains only the average price of each type of book.
- A subset of another view, or of some combination of views and base tables.

If you want to prevent some users from accessing the columns in the `titles` table that display money and sales amounts, you can create a view of the `titles` table that omits those columns, then give all users permission on the view but only the Sales Department permission on the table:

```
grant all on bookview to public
grant all on titles to sales
```

An equivalent way of setting up these privilege conditions, without using a view, is to use the following statements:

```
grant all on titles to public
revoke select, update on titles (price, advance,
    total_sales)
from public
grant select, update on titles (price, advance,
    total_sales)
to sales
```

One possible problem with the second solution is that users not in the `sales` group who enter the `select * from titles` command might be surprised to see the message that includes the phrase:

```
permission denied
```

SAP ASE expands the asterisk into a list of all the columns in the `titles` table, and since permission on some of these columns has been revoked from nonsales users, access to these columns is denied. The error message lists the columns for which the user does not have access.

To see all the columns for which they do have permission, the nonsales users must name them explicitly. For this reason, creating a view and granting the appropriate permissions on it is a better solution.

You can also use views for context-sensitive protection. For example, you can create a view that gives a data entry clerk permission to access only those rows that he or she has added or updated. To do so, add a column to a table in which the user ID of the user entering each row is automatically recorded with a default. You can define this default in the `create table` statement, like this:

```
create table testtable
    (empid       int,
     startdate   datetime,
     username     varchar(30) default user)
```

Next, define a view that includes all the rows of the table where `uid` is the current user:

```
create view context_view
as
    select *
    from testtable
    where username = user_name()
with check option
```

The rows retrievable through this view depend on the identity of the person who issues the `select` command against the view. By adding `with check option` to the view definition, you make it impossible for any data entry clerk to falsify the information in the `username` column.

Related Information

Understanding Ownership Chains [page 194]

7.9.2 Using Stored Procedures as Security Mechanisms

If a stored procedure and all underlying objects are owned by the same user, that owner can grant users permission to use the procedure without granting permissions on the underlying objects.

For example, you might give a user permission to execute a stored procedure that updates a row-and-column subset of a specified table, even though that user does not have any other permissions on that table.

You can create a procedure using `execute as owner` or `execute as caller`, which checks runtime permissions, executes DDL, and resolves objects names on behalf of the owner or caller respectively.

Procedures defined with `execute as owner`, `execute as caller`, or with no `execute as` clause, can be nested inside procedures defined with `execute as owner` or `execute as caller`. Similarly procedures defined with `execute as owner`or `execute as caller` can be nested inside procedures defined without the `execute as` clause.

`set session authorization` statement is not allowed inside the procedure created with `execute as owner` even if the statement is in a nested procedure which is not defined as `execute as owner`.

For syntax see, `create procedure` in *Reference Manual: Commands*.

7.9.2.1 Roles and Stored Procedures

Use the `grant execute` command to grant execute permission on a stored procedure to all users who have been granted a specified role.

`revoke execute` removes this permission. But `grant execute` permission does not prevent users who do not have the specified role from being granted execute permission on the stored procedure.

For further security, you can restrict the use of a stored procedure by using the `has_role` system function within the procedure to guarantee that a procedure can be executed only by users who have a given role. `has_role` returns 1 if the user has a specific role (`sa_role`, `sso_role`, `oper_role`, or any user-defined role) and returns 0 if the user does not have that role. For example, here is a procedure that uses `has_role` to see if the user has the system administrator role:

```
create proc test_proc
as
if (has_role("sa_role") = 0)
begin
    print "You don't have the right role"
    return -1
end
else
    print "You have SA role"
    return 0
```

See "System Functions" in *Reference Manual: Building Blocks* for more information about `has_role`.

7.9.3 Understanding Ownership Chains

Views can depend on other views or tables. Procedures can depend on other procedures, views, or tables. These dependencies can be thought of as an ownership chain.

Typically, the owner of a view also owns its underlying objects (other views and tables), and the owner of a stored procedure owns all the procedures, tables, and views referenced by the procedure.

A view and its underlying objects are usually all in the same database, as are a stored procedure and all the objects it references; however, this is not required. If objects are in different databases, a user wanting to use the view or stored procedure must be a valid user or guest user in all of the databases containing the objects. This prevents users from accessing a database unless the database owner has authorized it.

When a user who has been granted `execute` permission on a procedure or view uses it, SAP ASE does not check permissions on any of the underlying objects if:

- These objects and the view or procedure are owned by the same user, and
- The user accessing the view or procedure is a valid user or guest user in each of the databases containing the underlying objects.

However, if all objects are not owned by the same user, SAP ASE checks object permissions when the ownership chain is broken. That is, if object A references object B, and B is not owned by the user who owns object A, SAP ASE checks the permissions for object B. In this way, the owner of the original data is allowed to retain control over who is authorized to access it.

Ordinarily, a user who creates a view needs to worry only about granting permissions on that view. For example, say Mary has created a view called `auview1` on the `authors` table, which she also owns. If Mary grants `select` permission to Sue on `auview1`, Sue is allowed to access it without checking permissions on `authors`.

However, a user who creates a view or stored procedure that depends on an object owned by another user must be aware that any permissions he or she grants depend on the permissions allowed by those other owners.

7.9.3.1 Example of Views and Ownership Chains

Examples of ownership chains and permission checks.

As example of an ownership chain, Joe creates a view called `auview2`, which depends on Mary's view `auview1`. Joe grants Sue `select` permission on `auview2`.

The ownership chain looks like this:

Sue's permission	Objects	Ownership	Checks
select	auview2	Joe	*Sue not owner* *Check permissions*
select	auview1	Mary	*Different owner* *Check permissions*
none	authors	Mary	*Same owner* *No permission check*

SAP ASE checks the permissions on `auview2` and `auview1`, and finds that Sue can use them. SAP ASE checks ownership on `auview1` and `authors` and finds that they have the same owner. Therefore, Sue can use `auview2`.

Taking this example a step further, suppose that Joe's view, `auview2`, depends on `auview1`, which depends on `authors`. Mary decides she likes Joe's `auview2` and creates `auview3` on top of it. Both `auview1` and `authors` are owned by Mary.

The ownership chain looks like this:

Sue's permission	Objects	Ownership	Checks
select	auview2	Mary	*Sue not owner* *Check permissions*
select	auview2	Joe	*Different owner* *Check permissions*
select	auview1	Mary	*Different owner* *Check permissions*
none	authors	Mary	*Same owner* *No permission check*

When Sue tries to access `auview3`, SAP ASE checks permissions on `auview3`, `auview2`, and `auview1`. If Joe has granted permission to Sue on `auview2`, and Mary has granted her permission on `auview3` and

`auview1`, SAP ASE allows the access. SAP ASE checks permissions only if the object immediately before it in the chain has a different owner (or if it is the first object in the chain). For example, it checks `auview2` because the object before it—`auview3`—is owned by a different user. It does not check permission on `authors`, because the object that immediately depends on it, `auview1`, is owned by the same user.

7.9.3.2 Example of Procedures and Ownership Chains

Examples of ownership chains and permission checking for stored procedures.

Procedures follow the same rules as views. For example, suppose the ownership chain looks like this:

Sue's permission	Objects	Ownership	Checks
execute	proc4	Mary	*Sue not owner* *Check permissions*
none	proc3	Mary	*Same owner* *No permission check*
execute	proc2	Joe	*Different owner* *Check permissions*
execute	proc1	Mary	*Different owner* *Check permissions*
none	authors	Mary	*Same owner* *No permission check*

To execute `proc4`, Sue must have permission to execute `proc4`, `proc2`, and `proc1`. Permission to execute `proc3` is not necessary because `proc3` and `proc4` have the same owner.

Sue's permissions are checked on `proc4` and all objects it references each time she executes `proc4`. SAP ASE knows which referenced objects to check: it determined this the first time Sue executed `proc4`, and it saved the information with the procedure's execution plan. Unless one of the objects referenced by the procedure is dropped or redefined, SAP ASE does not change its initial decision about which objects to check.

This protection hierarchy allows every object's owner to fully control access to the object. Owners can control access to views and stored procedures, as well as to tables.

7.9.4 Permissions on Triggers

A trigger is a special kind of stored procedure used to enforce integrity, especially referential integrity. Triggers are never executed directly, but only as a side effect of modifying a table. You cannot `grant` or `revoke` permissions for triggers.

Only an object owner can create a trigger. However, the ownership chain can be broken if a trigger on a table references objects owned by different users. The protection hierarchy rules that apply to procedures also apply to triggers.

While the objects that a trigger affects are usually owned by the user who owns the trigger, you can write a trigger that modifies an object owned by another user. If this is the case, any users modifying your object in a way that activates the trigger must have permission on the other object as well.

If SAP ASE denies permission on a data modification command because a trigger affects an object for which the user does not have permission, the entire data modification transaction is rolled back.

See *Triggers: Enforcing Referential Integrity*, in the *Transact-SQL User's Guide*.

7.10 Execute a Procedure With execute as owner or execute as caller

You can create a procedure using `execute as owner` or `execute as caller`, which checks runtime permissions, executes DDL, and resolves objects names on behalf of the owner or caller respectively.

If you create a procedure using `execute as caller`, SAP ASE performs these operations as the procedure caller. If you create a procedure using `execute as owner`, these operations are performed on behalf of the procedure owner.

When the `execute as` clause is omitted, the behavior is the same as in versions earlier than SAP ASE 15.7 ESD #2.

Creating procedures for execution as the procedure owner is useful for applications that require all actions in a procedure to be checked against the privileges of the procedure owner. Implicit grant checking due to ownership chains does not apply to procedures created with execute as owner. The application end user requires no privilege in the database other than `execute` permission on the stored procedure. Additionally, any DDL executed by the procedure is conducted on behalf of the procedure owner, and any objects created in the procedure are owned by the procedure owner. This relieves the administrator of the requirement of having to grant privilege on DDL commands to the application user.

Creating the procedures for execution as the session user or caller is necessary if permissions must be checked on behalf of the individual user. For example, use `execute as caller` if a table accessed by the procedure is subject to fine-grained access control through predicated privileges, such that one user is entitled to see one set of rows and another user another set of rows. Implicit grant checking due to ownership chains does not apply for procedures which are created with execute as caller; if predicated privileges are present on tables referenced in the procedure, they will be applied.

If the `execute as` clause is omitted:

- Object names are resolved on behalf of the procedure owner.
- DDL commands and cross-database access are on behalf of the procedure caller.
- Permission checks for DML, `execute`, `transfer table`, `truncate table`, `delete statistics`, and `update statistics` are made on behalf of the caller unless there exists an ownership chain between the referenced object and the procedure, in which case permission checks are bypassed.

If the `execute as owner` clause is specified, the procedure behavior conforms to the expected behavior following an implicit `set session authorization` to the owner at the beginning of execution. This behavior includes:

- Object names are resolved on behalf of the procedure owner. If the procedure references a table or other object without qualifying the name with an owner name, SAP ASE will look up a table of that name

belonging to the procedure owner. If no such table exists, SAP ASE will look for a table of that name owned by the Database Owner.

- DDL commands and cross-database access are on behalf of the procedure owner.
- No implicit granting of permissions through ownership chains occurs.
- All access control checks are based on the permission of the procedure owner, and the owner's group, system roles, default user-defined roles, also those roles granted to the owner that are activated in the procedure body. Roles granted to the owner's login profile are also considered. Only those roles activated by the procedure owner during execution of the procedure are considered.
- Procedures defined with `execute as owner`, `execute as caller`, or with no execute as clause, can be nested inside procedures defined with `execute as owner`. Similarly procedures defined with `execute as owner` can be nested inside procedures defined without the `execute as` clause.
  Procedures called from an `execute as owner` procedure are executed as the owner of the calling procedure unless the nested procedure is defined as `execute as owner`.
- Dynamic SQL statements inside a procedure are executed with permissions of procedure owner regardless of the `'Dynamic Ownership Chain'` setting of `sp_procxmode`.
- Because temporary tables are owned by the session, temporary tables created outside the procedure by the caller are available inside the procedure to the procedure owner. This behavior reflects temporary table availability after a `set session authorization` command is executed in a session.
- Audit records for the execution of a procedure show the `execute as owner` clause and the name of the procedure owner in the extrainfo column.
- Audit records of statements executed within the procedure show the name of the procedure owner in the loginname column.
- `set session authorization` statement is not allowed inside the procedure created with `execute as owner` even if the statement is in a nested procedure which is not defined as `execute as owner`.

If the `execute as caller` clause is specified,

- Objects are resolved on behalf of caller. If the procedure references a table or other object without qualifying the name with an owner name, SAP ASE will look up a table of that name belonging to the user who called the procedure. If no such table exists, SAP ASE will look for a table of that name owned by the Database Owner.
- DDL commands and cross-database access are on behalf of the caller.
- No implicit granting of permissions through ownership chains occurs.
- Permissions are checked on behalf of caller, caller's group, active roles, and system roles.
- Procedures defined with `execute as owner`, `execute as caller`, or with no execute as clause, can be nested inside procedures defined with `execute as caller`. Similarly procedures defined with `execute as caller` can be nested inside procedures defined without the `execute as` clause.
  Procedures called from an `execute as caller` procedure are executed on behalf of the caller of the parent procedure unless the nested procedure is defined as `execute as owner`.
- Dynamic SQL executes as caller regardless of the `'Dynamic Ownership Chain'` setting on `sp_procxmode`.
- Temporary tables created outside the procedure are available inside the procedure.
- Object references by the procedure are not entered into `sysdepends`, as the objects are resolved according to each caller of the procedure.
- `select *` is not expanded in `syscomments`.
- Plans in the procedure cache for the same procedure are not shared across users, as the objects in the procedure must be resolved to the user executing the procedure. Because of this, procedure cache usage

may increase if many users are executing the procedure. The plan for a particular user is reused when the user executes the procedure again.

- Audit records for the execution of a procedure show the `execute as caller` clause and the name of the procedure caller in the extrainfo column.
- Audit records of statements executed within the procedure show the name of the session owner in the loginname column.

In the following example, the procedure created by user Jane has no `execute as` clause. The procedure selects from `jane.employee` into an intermediate table named `emp_interim`:

```
create procedure p_emp

    select * into emp_interim
    from jane.employee

grant execute on p_emp to bill
```

Bill executes the procedure:

```
exec jane.p_emp
```

- Bill is not required to have select permission on `jane.employee` because Jane owns `p_emp` and `employee`. By granting `execute` permission on `p_emp` to Bill, Jane has implicitly granted Bill `select` permission on `employee`.
- Bill must have been granted create table permission. The `emp_interim` table will be owned by Bill.

In the following example, Jane creates a procedure with an identical body using the `execute as owner` clause and Bill executes the procedure:

```
create procedure p_emp

    with execute as owner as

    select * into emp_interim
    from jane.employee

grant execute on p_emp to bill
```

- Bill requires only `execute` permission to run the procedure successfully.
- The `emp_interim` table is created on behalf of Jane, meaning Jane is the owner. If Jane does not have create table permission, the procedure will fail.

In the following example, Jane creates the same procedure with the `execute as caller` clause:

```
create procedure p_emp

    with execute as caller as

    select * into emp_interim
    from jane.employee
```

```
grant execute on p_emp to bill
```

- Bill must have `select` permission on `jane.employee`. No implicit grant checking is done though jane owns both `p_emp` and `employee`. If `jane.employee` has predicated privileges granted to Bill, the predicates will be added to the query.
- Bill must have create table permission. The `emp_interim` is created on behalf of Bill, meaning Bill is the owner.

7.10.1 Creating a Procedure with References to an Object with an Unqualified Name

Examples of creating procedures without the `execute as` clause, and with the `execute as owner` or `execute as caller` clause.

In the following example, the procedure does not have an `execute as` clause:

```
create procedure insert p
```

```
    insert t1 (c1) values (100)
```

```
grant execute on insert p to bill
```

Bill executes the procedure:

```
exec jane.insert p
```

- A search is performed to look for a table named `t1` owned by Jane. If `jane.t1` does not exist, a search is performed to look for `dbo.t1`.
- If `t1` resolves to `dbo.t1`, Bill must have insert permission for `t1`.
- If `t1` resolves to `jane.t1`, Bill will have implicit insert permission because of the ownership chain between `jane.insert_p` and `jane.t1`.

In the following example, Jane creates the same procedure as above with `execute as owner`:

```
create procedure insert p
```

```
    with execute as owner as
```

```
    insert t1 (c1) values (100)
```

```
grant execute on insert p to bill
```

Bill executes the procedure:

```
exec jane.insert p
```

- A search is performed to look for a table named `t1` owned by Jane. If `jane.t1` does not exist, a search is performed to look for `dbo.t1`.

- If `t1` resolves to `dbo.t1`, permission to insert into `t1` must have been granted to Jane.
- If `t1` resolves to `jane.t1`, since the procedure is being executed as owner, Jane has the permission.

In the following example, Jane creates the same procedure as above with `execute as caller`:

```
create procedure insert p
    with execute as caller as
    insert t1 (c1) values (100)
grant execute on insert p to bill
```

Bill executes the procedure:

```
exec jane.insert p
```

- A search is performed to look for a table named `t1` owned by Bill. If `bill.t1` does not exist, a search is performed to look for `dbo.t1`.
- If `t1` resolves to `dbo.t1`, Bill must have permission to insert into `t1`.

7.10.2 Procedures that Invoke a Nested Procedure in Another Database with a Fully Qualified Name

Examples of creating a nested procedure with `execute as owner` and `execute as caller`.

In the following example, Jane creates a procedure that invokes a nested procedure in another database with a fully qualified name. The login associated with Jane resolves to user Jane in `otherdb`. This example uses `execute as owner`:

```
create procedure p master
    with execute as owner
    as exec otherdb.jim.p_child
grant execute on p master to bill
```

Bill executes the procedure:

- SAP ASE checks that user Jane in `otherdb` has `execute` permission on `jim.p_child`.
- If `jim.p_child` has been created with no execute as clause, then `p_child` will be executed on behalf of Jane.
- If `jim.p_child` has been created with `execute as owner` then `p_child` will be executed on behalf of Jim.
- If `jim.p_child` has been created with `execute as caller` then `p_child` will execute on behalf of Jane.

In the following example, Jane creates the same procedure as above using `execute as caller`. The login associated with user Bill in the current database resolves to user Bill in `otherdb`:

```
create procedure p master
    with execute as caller
    as exec otherdb.jim.p_child
grant execute p master to bill
```

Bill executes the procedure:

```
exec jane.p_master
```

- SAP ASE checks that Bill in `otherdb` has `execute` permission on `jim.p_child`.
- If `jim.p_child` has been created with no execute as clause, then `p_child` will be executed on behalf of Bill.
- If `jim.p_child` has been created with `execute as owner` then `p_child` will be executed on behalf of Jim.
- If `jim.p_child` has been created with `execute as caller` then `p_child` will execute on behalf of Bill.

7.11 Row-Level Access Control

Row-level access control enables the database owner or table owner to create a secure data access environment automatically.

Row-level access control provides:

- More granular data security: you can set permissions for individual rows, not just tables and columns
- Automatic data filtering according to group, role, and application
- Data-level security encoded in the server

Row-level access control restricts access to data in a table's individual rows, through three features:

- Access rules that the database owner defines and binds to the table
- Application Context Facility, which provides built-in functions that define, store, and retrieve user-defined contexts
- Login triggers that the database owner, `sa_role`, or the user can create

SAP ASE enforces row-level access control for all data manipulation languages (DMLs), preventing users from bypassing the access control to get to the data.

The syntax for configuring your system for row-level access control is:

```
sp_configure "enable row level access", 1
```

This option slightly increases the amount of memory used, and you need an ASE_RLAC license option. Row-level access control is a dynamic option, so you need not restart the server.

Row-level access can also be granted using a `where` clause on the `grant` statement. Use this method of row-level access control if your privacy policy depends on data in other tables or on the full expression of SQL through a subquery.

7.11.1 Access Rules

To use the row-level access control feature, add the `access` option to the existing `create rule` syntax. Access rules restrict any rows that can be viewed or modified.

Access rules are similar to domain rules, which allow table owners to control the values users can insert or update on a column. The domain rule applies restrictions to added data, functioning on `update` and `insert` commands.

Access rules apply restrictions to retrieved data, enforced on `select, update`, and `delete` operations. SAP ASE enforces the access rules on all columns that are read by a query, even if the columns are not included in the select list. In other words, in a given query, SAP ASE enforces the domain rule on the table that is updated, and the access rule on all tables that are read.

For example:

```
insert into orders_table
select * from old_orders_table
```

In this query, if there are domain rules on the `orders_table` and access rules on the `old_orders_table`, SAP ASE enforces the domain rule on the orders_table, because it is updated, and the access rule on the `old_orders_table`, because it is read.

Using access rules is similar to using views, or using an ad hoc query with `where` clauses. The query is compiled and optimized after the access rules are attached, so it does not cause performance degradation. Access rules provide a virtual view of the table data, the view depending on the specific access rules bound to the columns.

Access rules can be bound to user-defined datatypes, defined with `sp_addtype`. SAP ASE enforces the access rule on user tables, which frees the table owner or database owner from the maintenance task of binding access rules to columns in the normalized schema. For instance, you can create a user-defined type, whose base type is `varchar(30)`, call it `username`, and bind an access rule to it. SAP ASE enforces the access rule on any tables in your application that have columns of type `username`.

7.11.1.1 Syntax for Access Rules

Use the `access` parameter in the `create rule` syntax to create access rules.

```
create [or|and] access rule (<access_rule_name>)
as (<condition>)
```

7.11.1.1.1 Creating a Sample Table with Access Rules

You can create a table and bind an access rule to the table.

Creating a Table

A table owner creates and populates table T (`username` char(30), `title` char(30), `classified_data` char(1024)):

```
AA, "Administrative Assistant","Memo to President"
AA, "Administrative Assistant","Tracking Stock                          Movements"
VP1, "Vice President", "Meeting Schedule"
VP2, "Vice President", "Meeting Schedule"
```

Creating and Binding Access Rules

The table owner creates access rule `uname_acc_rule` and binds it to the `username` column on table T.

```
create access rule uname_acc_rule
as @username = suser_name()
-----------
sp_bindrule uname_acc_rule, "T.username"
```

Querying the Table

When you issue the following query:

```
select * from T
```

SAP ASE processes the access rule that is bound to the `username` column on table T and attaches it to the query tree. The tree is then optimized and an execution plan is generated and executed, as though the user had executed the query with the filter clause given in the access rule. In other words, SAP ASE attaches the access rule and executes the query as:

```
select * from T where T.username = suser_name().
```

The condition `where T.username = suser_name()` is enforced by the server. The user cannot bypass the access rule.

The result of an Administrative Assistant executing the select query is:

```
AA, "Administrative Assistant","Memo to President"
AA, "Administrative Assistant","Tracking Stock Movements"
```

Dropping an Access Rule

Before you drop an access rule, you must unbind it from any columns or datatypes, using `sp_unbindrule`, as in the following example:

```
sp_unbindrule "T.username",
NULL, "all"
```

`sp_unbindrule` unbinds any domain rules attached to the column by default.

After you unbind the rule, you can drop it:

```
drop rule "rule_name"
```

For example:

```
drop rule "T.username"
```

7.11.1.1.2 Syntax for Extended Access Rule

Each access rule is bound to one column, but you can have multiple access rules in a table. `create rule` provides `AND` and `OR` parameters to handle evaluating multiple access rules.

To create `AND` access rules and `OR` access rules, use extended access rule syntax:

- `AND` access rule:

  ```
  create and access rule rule_name
  ```

- `OR` access rule

  ```
  create or access rule rule_name as
  ```

You can bind `AND` access rules and `OR` access rules to a column or user-defined datatype. With the extended access rule syntax, you can bind multiple access rules to the table, although you can bind only one per column. When the table is accessed, the access rules go into effect, the `AND` rules bound first by default, and then the `OR` access rules.

If you bind multiple access rules to a table without defining `AND` or `OR` access, the default access rule is `AND`.

If there is only one access rule on a row of the table and it is defined as an `OR` access rule, it behaves as an `AND` access rule.

7.11.1.2 Access and Extended Access Rules Examples

Examples for creating access rules.

Create Access Rules

The following steps create access rules:

```
create access rule empid1_access
as @empid = 1
```

```
create access rule deptno1_access
as @deptid = 2
```

The following steps create `OR` access rules:

```
create or access rule name1_access
as @name = "smith"
```

```
create or access rule phone_access
as @phone = "9999"
```

Create a Table

This step creates a test table:

```
create table testtab1 (empno int, deptno int,name char(10), phone char(4))
```

Bind Rules to Table

The following steps bind access rules to the test table columns:

```
sp_bindrule empid1_access, "testtab1.empno"
/*Rule bound to table column.*/
(return status = 0)sp_bindrule deptno1_access,"testtab1.deptno"
/*Rule bound to table column.*/(return status = 0)sp_bindrule
name1_access,"testtab1.name"
/*Rule bound to table column.*/(return status = 0)sp_bindrule
phone_access,"testtab1.phone"
/*Rule bound to table column.*/(return status = 0)
```

Insert Data into Table

The following steps insert values into the test table:

```
insert testtab1 values (1,1,"smith","3245")
(1 row affected)insert testtab1 values(2,1,"jones","0283")
(1 row affected)insert testtab1 values(1,2,"smith","8282")(1 row affected)insert
testtab1 values(2,2,"smith","9999")(1 row affected)
```

These examples show how access rules return specific rows containing information limited by access rules.

Example 1

Returns information from two rows:

```
/* return rows when empno = 1 and deptno = 2
and ( name = "smith" or phone = "9999" )
*/
select * from testtab1
 empno       deptno      name       phone
------------ ----------- ---------- -----
             1           2 smith      8282
             1           2 jones      9999
(2 rows affected)
/* unbind access rule from specific column */
sp_unbindrule "testtab1.empno",NULL,"accessrule"
/*Rule unbound from table column.*/
(return status = 0)
```

Example 2

Returns information from four rows:

```
/* return rows when deptno = 2 and ( name = "smith"
or phone = "9999" )*/select * from
testtab1 empno      deptno      name       phone
----------- ----------- ---------- -----
1           2           smith      8282
2           2           smith      9999
3           2           smith      8888
1           2           jones      9999
(4 rows affected)
/* unbind all deptno rules from specific column */
sp_unbindrule "testtab1.deptno",NULL,"all"
/*Rule unbound from table column.*/
(return status = 0)
```

Example 3

Returns information from six rows:

```
/* return the rows when name = "smith" or phone = "9999" */
select * from testtab1
 empno       deptno      name       phone
 ----------- ----------- ---------- -----
           1           1 smith      3245
           1           2 smith      8282
           2           2 smith      9999
           3           2 smith      8888
           1           2 jones      9999
           2           3 jones      9999
```

7.11.1.3 Access Rules and the alter table Command

When a table owner uses the `alter table` command, access rules are disabled during the execution of the command and enabled upon completion of the command. The access rules are disabled to avoid filtering the table data during the `alter table` command.

7.11.1.4 Access Rules and bcp

SAP ASE enforces access rules when data is copied out of a table using the `bcp`. SAP ASE cannot disable access rules, as it does with `alter table`, because any user can use `bcp` who has select permission on the table.

For security purposes, the database owner should lock the table exclusively and disable access rules during bulk copy out. The lock disables access to other users while the access rules are disabled. The database owner should bind the access rules and unlock the table after the data has been copied.

7.11.1.5 Access Rules as User-Defined Java Functions

Access rules can use user-defined Java functions. For example, you can use Java functions to write sophisticated rules using the profile of the application, the user logged in to the application, and the roles that the user is currently assigned for the application.

The following Java class uses the method `GetSecVal` to demonstrate how you can use Java methods that use JDBC as user-defined functions inside access rules:

```
import java.sql.*;
import java.util.*;
public class sec_class {
static String _url = "jdbc:sybase:asejdbc";
public static int GetSecVal(int c1)
{
try
{
PreparedStatement pstmt;
ResultSet rs = null;
Connection con = null;
    int pno_val;
pstmt = null;
Class.forName("sybase.asejdbc.ASEDriver");
con = DriverManager.getConnection(_url);
if (con == null)
{
return (-1);
}
pstmt = con.prepareStatement("select classification from sec_tab where id = ?");
if (pstmt == null)
{
return (-1);
}
pstmt.setInt(1, c1);
rs = pstmt.executeQuery();
rs.next();
pno_val = rs.getInt(1);
```

```
rs.close();
pstmt.close();
con.close();
return (pno_val);
}
catch (SQLException sqe)
{
return(sqe.getErrorCode());
}
catch (ClassNotFoundException e)
{
System.out.println("Unexpected exception : " + e.toString());
System.out.println("\nThis error usually indicates that " + "your Java CLASSPATH
environment has not been set properly.");
e.printStackTrace();
return (-1);
}
catch (Exception e)
{
System.out.println("Unexpected exception : " + e.toString());
e.printStackTrace();
return (-1);
}
}
}
```

After compiling the Java code, you can run the same program from `isql`, as follows.

For example:

```
javac sec_class.java
jar cufo sec_class. jar sec_class.class
installjava -Usa -Password -f/work/work/FGAC/sec_class.jar -
-D testdb
```

From `isql`:

```
/*to create new user datatype class_level*/
sp_addtype class_level, int
/*to create the sample secure data table*/
create table sec_data (c1 varchar(30),
c2 varchar(30),
c3 varchar(30),
clevel class_level)
/*to create the classification table for each user*/
create table sec_tab (userid int, clevel class-level int)
insert into sec_tab values (1,10)
insert into sec_tab values (2,9)
insert into sec_tab values (3,7)
insert into sec_tab values (4,7)
insert into sec_tab values (5,4)
insert into sec_tab values (6,4)
insert into sec_tab values (7,4)
declare @v1 int
select @v1 = 5
while @v1 > 0
begin
insert into sec_data values('8', 'aaaaaaaaaa', 'aaaaaaaaaa', 8)
insert into sec_data values('7', 'aaaaaaaaaa', 'aaaaaaaaaa', 7)
insert into sec_data values('5', 'aaaaaaaaaa', 'aaaaaaaaaa', 5)
insert into sec_data values('5', 'aaaaaaaaaa', 'aaaaaaaaaa', 5)
insert into sec_data values('2', 'aaaaaaaaaa', 'aaaaaaaaaa', 2)
insert into sec_data values('3', 'aaaaaaaaaa', 'aaaaaaaaaa', 3)
select @v1 = @v1 -1
end
go
```

```
create access rule clevel_rule
@clevel <= sec_class.GetSecVal(suser_id())
go
create default clevel_def as sec_class.GetSecVal(suser_id())
go
sp_bindefault clevel_def, class_level
go
sp_bindrule clevel, class_level
go
grant all on sec_data to public
go
grant all on sec_tab to public
go
```

7.11.2 Using the Application Context Facility

Applications on a database server must limit access to the data. Applications are carefully coded to consider the profile of the user. For example, a Human Resources application is coded to know which users are allowed to update salary information.

The attributes that enable this coding comprise an application context. The Application Context Facility (ACF) consists of three built-in functions that provide a secure environment for data access, by allowing access rules to compare against the intrinsic values assigned to users in a session.

An application context consists of context_name, attribute_name, and attribute_value. Users define the context name, the attributes, and the values for each context. You can use the default application context that SAP provides, `SYS_SESSION`, to access some session-specific information. All `SYS_SESSION` attributes are read-only except the attribute `INFMSG`.

The user profile, combined with the application profile, which is defined in a table created by the system administrator, permits cumulative and overlapping security schemes.

ACF allows users to define, store, and retrieve:

- User profiles (the roles authorized to a user and the groups to which the user belongs)
- Application profiles currently in use

Any number of application contexts per session are possible, and any context can define any number of attribute/value pairs. ACF context rows are specific to a session, and not persistent across sessions; however, unlike local variables, they are available across nested levels of statement execution. ACF provides built-in functions that set, get, list, and remove these context rows.

7.11.2.1 Setting Permissions for Using Application Context Functions

You execute an application context function in a select statement. The owner of the function is the system administrator of the server.

You can create, set, retrieve, and remove application contexts using these built-in functions.

- `set_appcontext` sets an application context name, attribute name, and attribute value, defined by the attributes of an application, for a specified user session.

- `get_appcontext` sets an application context name, attribute name, and attribute value, defined by the attributes of an application, for a specified user session.
- `list_appcontext` lists all the attributes of all the contexts in the current session.
- `rm_appcontext` removes a specific application context, or all application contexts.

For syntax, see *Reference Manual: Building Blocks*.

The data used in the functions is defined in a table that contains all logins for all tables, which created by the system administrator.

- `set_appcontext()` stores:

```
select set_appcontext ("titles", "rlac", "1")
```

- `get_appcontext()` supplies two parts of a context in a session, and retrieves the third:

```
select get_appcontext ("titles", "rlac")
```

```
------------------------
1
```

Granting and Revoking

Grant and revoke privileges to users, roles, and groups in a given database to access objects in that database. The only exceptions are `create database`, `set session authorization`, and `connect`. A user granted these privileges should be a valid user in the master database. To use other privileges, the user must be a valid user in the database where the object is located.

Using of functions means that unless special arrangements are made, any logged-in user can reset the profiles of the session. Although SAP ASE audits built-in functions, security may be compromised before the problem is noticed. To restrict access to these built-in functions, use `grant` and `revoke` privileges. Only users with the `sa_role` can grant or revoke privileges on the built-in functions. Only the `select` privilege is checked as part of the server-enforced data access control checks performed by the functions.

Valid Users

Because built-in functions do not have object IDs or home databases, database owners must grant `select` privileges for functions to the appropriate user or role. SAP ASE uses the user's default database as a template to determine privileges for additional databases. However, only the owner of the users' default database may grant `select` privileges. If using a built-in function in one databases prevents a user from using an additional database, the owner of the additional databases must explicitly revoke permission from the user or role in these databases.

Privileges granted to `public` affect only users named in the table created by the system administrator. Guest users have privileges only if the `sa_role` specifically grants it by adding them to the table.

A system administrator can execute the following commands to grant or revoke `select` privileges on specific application context functions:

- `grant select on set_appcontext to user_role`
- `grant select on set_appcontext to joe_user`
- `revoke select on set_appcontext from joe_user`

7.11.3 SYS_SESSION System Application Context

The `SYS_SESSION` context shows the default predefined application context, which provides session-specific pairs of attributes and values.

The syntax for using the context is:

```
select list_appcontext ("SYS_SESSION", "*")
```

Then:

```
select get_appcontext ("SYS_SESSION", "<attribute>")
```

Table 14: SYS_SESSION Attributes and Values

Attribute	Value
`username`	Login name
`hostname`	Host name from which the client has connected
`applname`	Name of the application as set by the client
`suserid`	User ID of the user in the current database
`groupid`	Group ID of the user in the current database
`dbid`	ID of the user's current database
`dbname`	Current database
`spid`	Server process ID
`proxy_suserid`	The server user ID of the proxy
`client_name`	Client name set by the middle-tier application, using `set client_name`
`client_applname`	Client application name set by the middle-tier application, using `set client_applname`
`client_hostname`	Client host name set by the middle-tier application, using `set client_hostname`

Attribute	Value
language	Current language the client is using by default or after using set language (@@language)
character_set	Character set the client is using (@@client_csname)
dateformat	Date expected by the client, set using set dateformat
is_showplan_on	Returns YES if set showplan is on, NO if it is off
is_noexec_on	Returns YES if set no exec is on, NO if it is off
infmsg	Returns a string with the latest informational messages sent to the client.

7.11.3.1 Solving a Problem Using an Access Rule and ACF

Problem solutions using Access Rule and ACF.

This shows the solution of a problem: each of five users, on different security levels, should see only rows with a value less than or equal to his or her security level. This solution uses access rules, with the Application Context Facility, to display only the rows that one of the users, Dave, sees.

There are five logins:

- Anne has security level 1.
- Bob has security level 1.
- Cassie has security level 2.
- Dave has security level 2.
- Ellie has security level 4.

Users should see only rows with a value in rlac that is less than or equal to their own security level. To accomplish this, create an access rule and apply ACF.

The rlac column is type integer, and appcontext arguments are type char.

```
create access rule rlac_rule as
  @value <= convert(int, get_appcontext("titles",
        "rlac"))
sp_bindrule rlac_rule, "titles.rlac"
/* log in as Dave and apply ACF value of 2*/
select set_appcontext("titles", "rlac", "2")
/*this value persists throughout the session*/
/*select all rows*/
select title_id, rlac from titles
---------------------
```

```
title_id     rlac
------------ --------
PC8888        1
BU1032        2
PS7777        1
PS3333        1
BU1111        2
```

```
PC1035        1
BU2075        2
PS2091        1
PS2106        1
BU7832        2
PS1372        1
(11 rows affected)
```

7.11.3.2 Using infmsg to Return Informational Messages

`infmsg` is an attribute in the `SYS_SESSION` system application context that returns the latest informational messages sent to the client as a string.

`infmsg` informational messages use a varchar (16384) data type. Text which exceeds 16384 bytes is truncated. `infmsg` is the only attribute in the `SYS_SESSION` context that is writable.

To start recording the printed informational messages, initialize the `infmsg` attribute:

```
SELECT set_appcontext("SYS_SESSION","INFMSG","")
```

To append a string to the end of the recorded message, execute the following command:

```
SELECT set_appcontext("SYS_SESSION","INFMSG","<string>")
```

To retrieve messages stored by the `infmsg` attribute, execute the following command:

```
SELECT get_appcontext("SYS_SESSION","INFMSG")
```

To clear the messages, execute the following command:

```
SELECT rm_appcontext("SYS_SESSION","INFMSG")
```

Example 1

This example runs a script called `database model` that dumps a database schema to file, and then loads the file to create a new copy of the database. By initializing the `infmsg` attribute between the dump and load commands, you can capture and review the informational messages generated:

```
DUMP database model to "/tmp/model.dmp"
GO

SELECT set_appcontext("SYS_SESSION","INFMSG","")
GO

LOAD DATABASE MODEL WITH listonly=(create_sql,load_sql)
GO
SELECT get_appcontext("SYS_SESSION","INFMSG")
GO
```

The returned result is:

```
 DISK INIT
```

```
         name = 'master'
        , physname = '/linuxstore3_work2/cruiz/ase160sp03plx//devices/
master_2k.dat'
        , size = '200M'
 go
 CREATE  DATABASE model
         ON master = '3M'
 FOR load
 go
 LOAD DATABASE model FROM '/tmp/model.dmp'
 go
 (1 row affected)
```

Example 2

This example initializes the `infmsg` attribute to return informational messages and then appends the strings `abc` and `pqr` to the end of the returned message:

```
select set_appcontext("SYS_SESSION", "INFMSG", "")
-----------
          0

(1 row affected)
```

```
select set_appcontext("SYS_SESSION", "INFMSG", "abc")
-----------
          0

(1 row affected)
```

```
select set_appcontext("SYS_SESSION", "INFMSG", "pqr")
-----------
          0

(1 row affected)
```

The returned result of the informational message with the appended text is:

```
select get_appcontext("SYS_SESSION", "INFMSG")
------
abcpqr

(1 row affected)
```

Example 3

Clear the recorded messages:

```
select rm_appcontext("SYS_SESSION", "INFMSG")
-----------
          0

(1 row affected)
```

Once cleared, no text is returned when you get the informational message:

```
select get_appcontext("SYS_SESSION", "INFMSG")
-----
NULL




(1 row affected)
```

7.11.4 Session Management for SAP HANA Syntax

Set variables for your database session using the SAP HANA session management style.

- Use the `set` command to define a session variable. For example:

  ```
  set 'abc' = '123'
  ```

  Use the `set` command to rewrite a session variable. For example:

  ```
  set 'abc' = '456'
  ```

- Use `unset` to delete a session variable. For example:

  ```
  unset 'abc'
  ```

- Use the `session_context` function to retrieve a variable. For example:

  ```
  select session_context('abc')
  ```

7.11.5 Using Login Triggers

Login triggers execute a specified stored procedure every time a user logs in.

The login trigger is an ordinary stored procedure, except it executes in the background. It is the last step in a successful login process, and sets the application context for the user logging in.

Only the system security officer can register a login trigger to users in the server.

To provide a secure environment, the system administrator must:

1. Revoke `select` privilege on the `set_appcontext` function. The owner of a login trigger must have explicit permission to use `set_appcontext`, even if the owner has `sa_role`.
2. Configure a login trigger from a stored procedure for each user, and register the login trigger to the user.
3. Provide execute privilege to the login trigger that the user executes.

7.11.5.1 Create Login Triggers

Create a login trigger as a stored procedure.

Do not use the `create trigger` command. The following sample requires the you first create the `lookup` table in the `pubs2` database:

```
create table lookup (
  appname varchar(20),
  attr varchar(20),
  value varchar(20),
  login varchar(20)
)
```

Then create a login trigger stored procedure in the `pubs2` database:

```
create procedure loginproc as
  declare @appnamevarchar(20)
  declare @attr varchar(20)
  declare @value varchar(20)
  declare @retvalue int
declare apctx cursor for
select appname, attr, value from
pubs2.dbo.lookup where login = suser_name()
open apctx
fetch apctx into @appname, @attr, @value
While (@@sqlstatus = 0)
  begin
    select f@retval =
      set_appcontext (rtrim (@appname),
      rtrim(@attr), rtrim(@value))
  fetch apctx into @appname, @attr, @value
 end
go
```

Grant permission to execute loginproc to public:

```
grant execute on loginproc to public
```

To associate the login trigger with a specific user, run `alter login` in the user's default database.

7.11.5.2 Configure Login Triggers

You must have `sso_role` enabled to set, change, or drop a login trigger.

The object ID of the login trigger is stored in the `syslogins.procid` column. Login triggers do not exist by default. They must be registered using `alter login`.

Run this command from the user's default database. The stored procedure you are registering as a login trigger must be available in the user's default database, because SAP ASE searches the `sysobjects` table in the user's default database to find the login trigger object.

The following example configures the stored procedure `my_proc` (which must exist in the database you want to configure) as a login trigger for login `my_login`:

```
alter login my_login modify login script "my_proc"
```

Again, you must execute the command from within the user's default database. SAP ASE checks to see whether the login has `execute` permissions on the stored procedure, but not until the user actually logs in and executes the login trigger.

7.11.5.2.1 Display Login Triggers

To display the current login trigger, use `sp_displaylogin`.

```
sp_displaylogin my_login
go
(....)
Default Database: my_db
Default Language:
Auto Login Script: my_proc
....
```

7.11.5.2.2 Drop and Change Login Triggers

Once you have configured a stored procedure as a login trigger, it cannot be dropped. You must first unconfigure the stored procedure, either by dropping the login trigger altogether, or by changing the login trigger to a different stored procedure.

To drop the login trigger, enter:

```
alter login my_login drop login script
```

To change the login trigger to a different stored procedure, enter:

```
alter login my_login modify login script "diff_proc"
```

7.11.5.3 Execute Login Triggers

Login triggers are different from ordinary stored procedures in that once they are registered they execute in the background, without active user connections.

Once you have configured a login trigger, SAP ASE automatically executes it in the background as soon as the user logs in, but before the server executes any commands from the client application.

If one login makes multiple concurrent connections, the login trigger executes independently during each session. Similarly, multiple logins can configure the same stored procedure to be a login trigger.

Background execution means that you cannot use some standard features of stored procedures in a stored procedure configured as a login trigger. For instance, you cannot pass any parameters without default values to or from the procedure, nor does the procedure pass back any result values.

This special execution mode affects any stored procedures that are called by the login trigger stored procedure, as well as any output generated by the login trigger stored procedure itself.

You can also execute a login trigger stored procedure as a normal stored procedure, for example, from `isql`. The procedure executes and behaves normally, showing all output and error messages as usual.

7.11.5.4 Login Trigger Output

The main effect of executing the stored procedure as a background task is that output from the login trigger is not written to the client application, but to the error log file, as are some, but not all, error messages.

Output from `print` or `raiserror` messages is prefixed by the words `background task message` or `background task error` in the error log. For example, the statements `print "Hello!"` and `raiserror 123456` in a login trigger appear in the error log as:

```
(....) background task message: Hello!
(....) background task error 123456: This is test
message 123456
```

However, not all output goes to the error log:

- No result sets from `select` statements (which are normally sent to a client connection) appear anywhere, not even in the error log. This information disappears.
- The following statements execute normally: `insert...select` and `select...into` statements, as well as other DML statements which do not ordinarily send a result set to the client application, and DDL statements ordinarily allowed in a stored procedure.

7.11.5.5 Use Login Triggers for Other Applications

Login triggers are part of the row-level access control feature in SAP ASE.

In this context, you can use a login trigger in combination with the features for access rules and application contexts to set up row-level access controls, once a session logs in to an SAP ASE server. However, you can use login triggers for other purposes as well.

7.11.5.5.1 Limiting the Number of Concurrent Connections

You can create a stored procedure to limit the number of concurrent connections.

Procedure

1. As system administrator, create the `limit_user_sessions` stored procedure:

```
create procedure limit_user_sessions
as
  declare @cnt int,
```

```
   @limit int,
   @loginname varchar(32)
  select @limit = 2 -- max nr. of concurrent logins
/* determine current #sessions */
select @cnt = count(*)
from master.dbo.sysprocesses
  where suid = suser_id()
/* check the limit */
if @cnt > @limit
begin
  select @loginname = suser_name()
  print "Aborting login [%1!]: exceeds session
    limit [%2!]",
      @loginname, @limit
    /* abort this session */
    select syb_quit()
end
go
grant exec on limit_user_sessions to public
go
```

2. As system security officer, configure this stored procedure as a login trigger for user “bob”:

```
alter login bob modify login script
"limit_user_sessions"
go
```

Now, when user “bob” creates a third session for SAP ASE, this session is terminated by the login trigger calling the `syb_quit()` function:

```
% isql -SASE125 -Ubob -Pbobpassword
1> select 1
2> goCT-LIBRARY error:
ct_results(): network packet layer: internal net library error:
Net-Library operation terminated due to disconnect
```

This message appears in the error log file:

```
(...) background task message: Aborting login
[ my_login]: exceeds session limit [2]
```

7.11.5.5.2 Enforcing Timed-Based Restrictions

You can limit the number of concurrent connections for a specific login and restrict access to specific times of day for that login.

Procedure

1. As system administrator, create this table:

```
create table access_times (
suid int not null,
dayofweek tinyint,
shiftstart time,
shiftend time)
```

2. As system administrator, insert the following rows in table `access_times`. These rows indicate that user "bob" is allowed to log into the server on Mondays between 9:00am and 5:00pm, and user "mark" is allowed to login to the server on Tuesdays between 9:00am and 5:00pm.

```
insert into access_times
select suser_id('bob'), 1, '9:00', '17:00'
go
insert into access_times
select suser_id('mark'), 2, '9:00', '17:00'
go
```

3. As system administrator, create the `limit_access_time` stored procedure, which references the `access_time` table to determine if login access should be granted:

```
create procedure limit_access_time as
declare @curdate date,
    @curdow tinyint,
    @curtime time,
    @cnt int,
    @loginname varchar(32)
-- setup variables for current day-of-week, time
select @curdate = current_date()
select @curdow = datepart(cdw,@curdate)
select @curtime = current_time()
select @cnt = 0
-- determine if current user is allowed access
select @cnt = count(*)
from access_times
where suid = suser_id()
and dayofweek = @curdow
and @curtime between shiftstart and shiftend
if @cnt = 0
begin
   select @loginname = suser_name()
   print "Aborting login [%1!]: login attempt past      normal working
hours", @loginname
   -- abort this session
   return -4
end
go
grant exec on limit_access_time to public
go
```

4. As system security officer, configure the `limit_access_time` stored procedure as a login trigger for users "bob" and "mark":

```
alter login bob login script
"limit_access_time"
go
alter login mark login script
"limit_access_time"
go
```

   On Mondays, user "bob" creates a session:

```
isql -Ubob -Ppassword
1> select 1
2> go
-----------
          1
(1 row affected)
```

However, user “mark” is denied access:

```
isql -Umark -Ppassword
1> select 1
2> go
CT-LIBRARY error:
ct_results(): network packet layer: internal net
library error: Net-Library operation terminated
due to disconnect
```

The following message is logged in the error log:

```
(...) server back-ground task message: Aborting
login [mark]: login attempt past normal working
hours
```

Results

The above examples show how you can limit the number of concurrent connections for a specific login and restrict access to specific times of day for that login, but it has one disadvantage: the client application cannot easily detect the reason the session was terminated. To display a message to the user, such as “Too many users right now—please try later,” use a different approach.Instead of calling the built-in function `syb_quit()`, which causes the server to simply terminate the current session, you can deliberately cause an error in the stored procedure to abort the login trigger stored procedure.

For example, dividing by zero aborts the login trigger stored procedure, terminates the session, and causes a message to appear.

7.11.5.6 Login Trigger Restrictions

These login trigger actions are restricted.

- You cannot create `#temp` tables to use later in the session. Once the procedure completes, `#temp` tables are automatically dropped and the original session settings are restored, as in any other stored procedure.
- Do not use login triggers on the `sa` login; a failing login trigger can lock you out of SAP ASE.
- Do not use a login trigger for anything that may take longer than a few seconds to process, or that risks processing problems.

Issues and Information

- If you do not have access to the error log, do not use login triggers. Always check the error log for error messages.
- For SAP ASE version 15.0.2 and later, any exportable option set or unset in a login trigger take effect in the login process when the server starts.
  To disable this behavior, execute `set export_options off` inside the login trigger.

SAP ASE versions 15.0.1, 12.5.4, and earlier required that you start the server with trace flag 4073 to enable the options for a login trigger.

- A client application, like `isql`, is unaware of the existence or execution of a login trigger; it presents a command prompt immediately after the successful login, though SAP ASE does not execute any commands before the login trigger successfully executes. This `isql` prompt displays even if the login trigger has terminated the user connection.
- The user logging in to the server must have `execute` permission to use the login trigger stored procedure. If no `execute` permission has been granted, an error message appears in the error log and the user connection closes immediately (though `isql` still shows a command prompt).
  The error log shows a message similar to the following:

  ```
  EXECUTE permission denied on object my_proc,
  database my_db, owner dbo
  ```

- The login trigger stored procedure cannot contain parameters without specified default values. If parameters without default values appear in the stored procedure, the login trigger fails and an error similar to the following appears in the error log:

  ```
  Procedure my_proc expects parameter @param1, which
  was not supplied...
  ```

7.11.5.7 Disable the execute Privilege on Login Triggers

A database owner or administrator can disable `execute` privilege on the login trigger, or code the login trigger to permit access only at certain times.

For example, you may want to prohibit regular users from using the server while the database owner or administrator is updating the table.

> **i** Note
>
> If the login trigger returns a negative number, the login fails.

7.11.6 Export set Options from a Login Trigger

SAP ASE allows options for the `set` command that are inside login triggers to remain valid for the entire user session.

The following `set` options are automatically exported:

- `showplan`
- `arithabort [overflow | numeric_truncation]`
- `arithignore [overflow]`
- `colnames`
- `format`
- `statistics io`

- procid
- rowcount
- altnames
- nocount
- quoted_identifier
- forceplan
- fmtonly
- close on endtran
- fipsflagger
- self_recursion
- ansinull
- dup_in_subquery
- or_strategy
- flushmessage
- ansi_permissions
- string_rtruncation
- prefetch
- triggers
- replication
- sort_resources
- transactional_rpc
- cis_rpc_handling
- strict_dtm_enforcement
- raw_object_serialization
- textptr_parameters
- remote_indexes
- explicit_transaction_required
- statement_cache
- command_status_reporting
- proc_return_status
- proc_output_params

7.11.7 Set Global Login Triggers

Use `sp_logintrigger` to set a global login trigger that is executed at each user login.

To take user-specific actions, set a user specific login trigger using `alter login` or `create login`.

8 Predicated Privileges

Predicated privileges include a system of flexible row-level access controls, allowing you to grant select, update, and delete privileges on data for different users, groups, or roles based on a predicate that is evaluated by SAP ASE during data access.

If the condition expressed by the predicate is not met for any row of data, that row is withheld from the result set.

Predicated privileges offer data privacy protection based on row-level access controls that dynamically grant privileges to a user based on data content or context information, allowing you to implement a privacy policy in the server instead of the client or a Web server.

A predicate may access other objects, such as tables, SQL functions, or built-in functions. Access to these other objects is checked against the permissions and roles of the predicate owner (the grantor of the predicated privilege). The user who executes the `select`, `update`, or `delete` command on the object targeted by the grant does not require explicit permission on the objects referenced by the predicate.

Predicated privileges allow a service provider to store data in a single database, and share the same tables for multiple customers instead of requiring separate views and `instead of` triggers for each customer. The data is filtered for specific users according to the predicated privileges granted directly to the user, or granted indirectly through the user's groups or roles.

8.1 Differences Between Access Rules and Predicated Privileges

Compare the advantages of predicated privileges over the access rules provided by versions of SAP ASE earlier than 15.7 ESD#2.

	Access rules	Predicated privileges
Predicate scope	Can refer only to the column to which the rule is bound. Multiple access rules are required to reference more than one column.	Can refer to any column in the table or, using a sub-select, any other table or function in the current database or in other databases.
Statement scope	All rules apply to `select`, `update`, and `delete`. Does not allow different restrictions across different kinds of access.	Grantor specifies which predicate applies to which access. For example, stricter access control may be imposed for updating and deleting rows than for selecting rows.

	Access rules	Predicated privileges
Combining predicates	Gives a way to combine individual rules using `and` or `or`, but does not give a way to express precedence grouping; for example, `((<rule1> and <rule2>) or <rule3>)`.	A predicated privilege can construct an arbitrarily complex Boolean expression. You may combine multiple privileges against the same object based on well-defined rules.
Restricting the scope to a subject	Any restriction directed at a specific subject must be expressed as part of the rule, such as `where name = <user>`.	You can apply a predicate to selected users, groups, or roles without the administrator needing to introduce complicated logic into the predicate.
Integration with permissions system	Decoupled from SQL authorization. Requires rule analysis to understand how users are restricted from some action.	Strongly enforced through the permission system. Diagnostic commands allow an application developer to predict how the predicates modify a query .

8.2 Commands Used for Predicated Privileges

Use these commands for granting and revoking predicated privileges.

- `grant ... where` – grant row-level access control to a group, role, or individual user based on conditions expressed through the `where` clause.
- `grant role ... where` – grant roles whose activation is conditional on the evaluation of a `where` clause
- `revoke ... [with { <pred_name> | {all |no} predicates}]` – revoke predicated privileges.
- `set show_transformed_sql` – displays the full SQL text, including the added predicates for row-level filtering.

See the *Reference Manual: Commands*.

8.3 Configure SAP ASE to Use Predicated Privileges

Configure SAP ASE to use predicated privileges.

1. Enable the ASE_SECDIR license, required for SAP ASE Security and Directory Services.
2. Configure the `enable predicated privileges` configuration parameter.
3. Set the value for `permission cache entries`. Granting access to a single table restricted by multiple predicated privileges increases the number of protection cache entries required. Allow one extra protection cache entry for any access that requires more than one predicate evaluation.
4. Consider whether to increase the size of the procedure cache. If users executing a stored procedure each have different predicated access to the tables referenced by the procedure, a separate plan is created for each user executing the procedure. In this case, you may need to increase the size of the procedure cache.

8.4 Grant Predicated Privileges

`grant` allows you to grant row-level access control on a table based on conditions specified by a `where` clause.

The syntax is:

```
grant {all [privileges] | permission_list}
on table_name [<correlation_name>]
[(<column_name_list>)]
[where search_conditions
   [as pred_name]]
to {public | <name_list> | <role_list>}
[with grant option]
```

See the *Reference Manual: Commands*.

The `grant ... where` statement on a table allows you to define conditions on `select`, `update`, or `delete` commands against that table that qualify or disqualify rows from the result set. The `<search_conditions>` act as a row filter, working with the `where` clause specified on the `select`, `update`, or `delete`. You can compose `search_conditions` that contain complex SQL conditions and reference functions, tables, and SQL functions. However, predicated privileges cannot reference a view.

This example gives engineers permission to see their own salary and the salary of the workers reporting to them:

```
grant select on employee (salary)
  where ename = USER or
  emgr = USER to eng_role
```

The statement filters rows on the `select` command (including `select`s that are part of compound commands such as `select into`, `insert select`, and `update from`, and `select` commands wrapped by a view), as well as on `select`, `update`, and `delete` commands that reference the `salary` column in their `where` clauses. The predicate, effectively, restricts workers from seeing another worker's salary.

SAP ASE applies the `where` clause from the `grant` statement when a query against the table is processed, and evaluates the conditions when a query is executed.

8.4.1 Grant Access to select Data

The `where` clause on a `grant select` command combines with any end user's `where` clause in the `select` command to qualify the rows returned from a `select` operation.

Use the `grant ... where` statement on a table without a column list to filter rows regardless of the individual columns selected. This `grant` statement allows `project_leaders` read and write access to information only for those projects they manage:

```
grant select on projects
  where user_id() = p_mgrid
  to proj_leader_role
```

Use a column or column list with the `grant` command to filter rows when the `select` accesses a specific column. These `grant` statements allows all engineers access to the names and departments of all engineers, while restricting access to engineers' phone numbers to their manager:

```
grant select on employee(ename, edept)
  to eng_role
```

```
grant select on employee(ephone)
  where user_id() = emgr to eng_role
```

8.4.2 Grant Access to update Data

The `where` clause on a `grant update` command combines with the end user's `where` clause in the `select` command to qualify the rows targeted for an `update` operation.

Use a column or column list with the `grant ... update` statement to filter rows that update only specific columns. This example allows all administrative and medical staff to update patients' addresses, but restricts modifications to the medical status to only the patient's physician:

```
grant update on patients (address, phone)
  to public
grant update on patients (medical_status)
  where USER = primary_md
  to doctor_role
```

The `update` command raises a permissions error if any user other than one with doctor_role attempts to update a patient's medical status. It also restricts the rows and columns that a doctor may update.

8.4.3 Grant Access to delete Data

The `where` clause on a `grant update` command combines with the end user's `where` clause to qualify the rows to be deleted.

This example grants permission for Bob to delete sales data:

```
grant delete on employee
  where edept = 'sales'
  to Bob
```

Now when Bob runs this statement:

```
delete employee
  where status = 'terminated'
```

SAP ASE runs the statement internally as:

```
delete employee
  where status = 'terminated' and edept = 'sales'
```

8.4.4 Use Predicated Privileges to Enforce a Data Privacy Policy

The `grant` statements can be used to enforce a policy that restricts employees from viewing private data.

The `grant` statements in this example enforce a policy that restricts employees from viewing any salaries but their own, allows managers to view their group members' salaries, and restricts salary updates to the HR department during any month except December:

```
grant select on employee as e (esalary)
  where e.ename = USER
  or USER in
   (select username(mgrid) from depts d
  where d.deid = e.eid)
  to public
grant update on employee (esalary)
  where datepart(month, current_date()) <> 12
  to hr_role
```

8.5 Revoke Predicated Privileges

Use the revoke command to revoke row-level access on a table.

The syntax is:

```
revoke {all [privileges]
    | [all] <permission_list>}
on <table_name> (<column_list>)
[with { <pred_name> | {all |no} predicates}]
from {public | <name_list> | <role_list>}
```

- If `<column_list>` is used with `<pred_name>`, the predicated row-level access is revoked for the named columns. If there are remaining columns that are referenced by this row-level privilege, the privilege and its related named predicate remain in `sysprotects`. For the following `revoke` examples, assume permission was initially granted as:

  ```
  grant select on t1(col1,col2,col3)
    where col4 > 99 as pred1
    to user1
  ```

  This revokes `select` permission on `t1.col2` with `pred1`:

  ```
  revoke select on t1 (col2)
    with pred1
    from user1
  ```

  However, if `user1` selects `t1.col3`, then `pred1` is still applied:
  If the grantor also issued:

  ```
  revoke select on t1 (col1, col3)
    with pred1
    from user1
  ```

or,

```
revoke select on t1 with pred1
```

then all permissions on `t1` using `pred1` would be revoked from `user1`.

- Use `with` cause to revoke all predicates, name predicates, or remove only unpredicated grants for the given access from the named grantee. If the `with` clause is omitted, both predicated and non-predicated grants are revoked. This example revokes all predicated row-level privileges granted to `user1`.

  ```
  revoke select on t1 with all predicates
    from user1
  ```

  The following omits the `with` clause and revokes all non-predicated `select` access non-predicated grants granted to `user1`.

  ```
  revoke select on t1 with no predicates
  ```

See the *Reference Manual: Commands*.

8.6 How SAP ASE Saves Predicated Privileges in sysprotects

For `grant` commands that do not include predicates, the precedence rule of granted privileges with respect to the grantee, specifies that a `grant` at a higher level can remove a `grant` at a lower level. However, if the higher precedence `grant` is predicated, generally lower-precedence rows are retained in `sysprotects`.

In this example, the second non-predicated `grant` to public causes SAP ASE to remove the earlier `grant` to an individual `user1`:

```
grant select on t1 to user1
grant select on t1 to public
```

However, grants with predicates allow grants that are lower in precedence to remain. For example, the first `grant`, below, to `user1` is general, but the second `grant` to `public` on `t1` is conditional. SAP ASE does not remove the `grant` to `user1`, even though the grantee, `public`, has a higher precedence:

```
grant select on t1
  to user1
grant select on t1
  where col1 = 4 to public
```

In this example, the second `grant` is added but the privilege is recorded as conditional access on all columns in `t1` except `col1`:

```
grant select on t1 (col1)
  to user2
grant select on t1
  where col1 = 4
  to user2
```

The first `grant` is not removed because it gives `user2` unconditional access on column `col1`, which is stronger than the conditional access on `col1` from the second `grant`.

8.7 Predicated Role Activation

Use the `grant role ... where` command to restrict role activation according to certain conditions.

Only active roles can have access privileges. The grantee of a role with an activation predicate may be a user or a login profile (a login profile is a template of attributes that is applied to assigned users when they log in).

The syntax is:

```
grant role <role_name>
        [where <pred_expression>]
    to {<username>< | ><login_profile_name >}
```

The `where <pred_expression>` clause is a role-activation predicate or SQL condition that must be satisfied when the named role is activated. Use `<pred_expression>` when granting a role to a user or login profile.

> **i Note**
>
> You must be in the `master` database to grant a role with an activation predicate.

Role activation predicates, like predicated privileges, can access database objects such as tables, views, SQL functions, and built-in functions. These accesses are checked against the permissions and roles of the predicate owner (the grantor of the role) instead of requiring explicit permission on the objects accessed by the predicate by the user who executes the set role statement.

> **i Note**
>
> Although you can reference a view with an activation predicate, you cannot reference a view with a row-filtering predicate.

The following example allows login "Bob" to perform maintenance duties on the server during off-peak hours. If Bob attempts to enable the `oper_role` between 8 am and 6 pm, an error is returned.

```
grant role oper_role
  where datepart(hour, current_time())
  not between 8 and 18
  to Bob
```

If an activation predicate is used on a `grant role` to a user or login profile, the predicate is evaluated either:

- During login, if the role is designated as a default role for the user, or
- During login, if the role is granted to a login profile and has been added as an auto-activated role, or
- When the user attempts to set the role.

Roles granted to a login profile apply to all users assigned to that profile, and can be specified for automatic activation at login, based on evaluation of the predicate. For example, this allows users associated with the `admin` login profile to assume the sa_role when they use the `resource_monitor` application:

```
grant role sa_role
```

```
  where
    (select get_appcontext('SYS_SESSION', 'applname'))
    = 'mon_resource'  to loginprof_admin
```

If a role activation predicate evaluates to false, an error is returned for the `set role` command, or silently does not set a default or automatically activated role during login.

8.8 Combine Predicates to Enforce Row-Level Privileges

SAP ASE combines and attaches predicates to `update`, `delete`, or `select` commands on a specific table.

This done while taking into account:

- Multiple grantors – `grant` commands are applied across all affected columns to a specific grantee, or to his group or roles, for the same access from one or more grantors.
- Multiple column sets – `grant` commands are applied across different column sets, to a specific grantee, or to his group or roles, for the same access from one or more grantors.
- Different access – granted privileges on the same object for different accesses.

If a `revoke` row exists for a specific user access at the user or group level, the aggregated grants consist of predicates that apply to the user's set of roles.

SAP ASE obeys a set of rules when combining predicates.

Examples

The following examples assume you create this table:

```
create table employee (
   eid int not null,
   ename char(8) not null,
   eaddr varchar(20) not null,
   ecity char(10) not null,
   ezip char(5) not null,
   ephone int not null,
   esalary money not null,
   edept char(10) not null
)
```

And insert these values:

```
insert employee
values (1001, 'Bill','6 Tree St','Baytown','97364',3973465,100000, 'eng')
insert employee
values (1009, 'Sally', '2 Sun Dr', 'Henton', '97821', 1239786, 50000, 'sales')
insert employee
values (1005, 'Drew','7 Inn Rd','Denville','29052',4960034,110000, 'payroll')
insert employee
values (1010, 'Navid', '1 Bat Ave', 'Busitown', '60734', 3094509,175000, 'hr')
insert employee
values (1004, 'Pat', '5 Dot Ln', 'Toytown', '33109', 4247184, 90000, 'eng')
insert employee
values (1006, 'Sue', '4 Tip St', 'Uptown', '76420', 5258641, 200000, 'sales')
```

8.8.1 Example 1: Combining Predicates From Multiple Grants for the Same Table Access

Predicates from multiple grants for the same table access are combined using a Boolean `or` operator when all predicates apply to the columns referenced by the user command.

If a user is directly, or indirectly, granted (for example, through a role) multiple predicated privileges for the same access and on the same object, the predicates from the individual `grant` commands are combined using the `or` parameter. Additional grants with predicates for a single access usually results in more rows being made visible to the user.

For example, user Ben, a sale manager, has been granted these permissions on the `employee` table:

```
grant select on employee
  where edept = "sales"
  to sales_mgr_role
grant select on employee (eid, ename, eaddr,
  ezip)
  where ezip like "97%"
  to ben
```

When Ben with `sales_manager_role` active, executes this statement, the columns `eid` and `ename` are affected by the two predicated `grant` commands:

```
select eid, ename from employee
```

SAP ASE internally executes:

```
select eid, ename from employee
  where edept = "sales" or
  ezip like "97%"
```

SAP ASE then returns:

```
eid       ename
-----      ------
1001      Bill
1009      Sally
1006      Sue
```

8.8.2 Example 2: Combining Predicates From Multiple Grants Applying to Different Sets of Columns

Predicates from multiple grants applying to different sets of columns are combined using the boolean `or` and `and` operators.

When a result set is affected non-uniformly by a set of predicates, SAP ASE combines all predicates for a given access on each column using the boolean `or` operator. These Boolean expressions are combined across all columns using the Boolean `and` operator.

For example, Ben has been granted the same privileges as in Example 1, except the first `grant` affects a subset of employee columns:

```
grant select on employee (esalary)
  where edept = "sales"
  to sales_mgr_role
grant select on employee (eid, ename, eaddr,
  ezip)
  where ezip like "97%"
  to ben
```

When Ben executes this statement, SAP ASE notices that columns `ename` and `esalary` are affected by the different `grant` commands:

```
select ename, esalary from employee
```

SAP ASE internally executes:

```
select ename, esalary from employee
  where edept = "sales" and ezip like "97%"
```

SAP ASE then returns:

```
ename     esalary
sally     50000
```

8.8.3 Example 3: Combining Predicates From Multiple Grants for Different Access

Predicates from multiple grants for different accesses are combined using the Boolean `and` operator.

If `select access` on an object is granted to a subject using a predicate, update access is allowed using a different predicate, and both predicated grants are required for access, the two predicates are combined using the Boolean `and` operator.

For example, the first `grant` allows California administrators to see employees who live in the "97..." area code. Through the second `grant`, those administrators are allowed to update only employees whose salary is less than $100,000 per year:

```
grant select on employee
  where ezip like "97%"
  to calif_admin
grant update on employees (esalary)
  where esalary < $100000
  to calif_admin
```

When a user with `calif_admin` role executes:

```
select eid, ename, esalary from employee
```

SAP ASE returns:

```
eid     ename     esalary
----     ------    -----------
```

```
1001       Bill     100,000.00
1009      Sally      50,000.00
```

When a user with `calif_admin` executes:

```
update employees
set esalary = esalary + (esalary * 0.05)
```

SAP ASE internally executes:

```
update employees
set esalary = esalary + (esalary * 0.05)
  where ezip like "97%"
  and esalary < $100000
```

SAP ASE then updates the row as:

```
eid     ename     esalary
1009    Sally      $52500
```

8.9 SQL Behavior with Predicated Privileges

Use these `set` parameters and system procedures to display information about predicated privileges.

These set parameters display information about predicated privileges:

- `set show_transformed_sql` command displays the full SQL query, including the added predicates for row-level filtering. When used with `set noexec on`, you can validate the final SQL query after the relevant predicates for the user, group, and active roles have been added, without executing the query.
- `set show_permission_source` displays the grantee, type of grantee, grantor, action, object and predicate in tabular form.

See the *Reference Manual: Commands*.

These system procedures display information about predicated privileges:

- `sp_helptext` displays the predicate text. Include the predicate's user-defined name, if there is one, or its internal name with `sp_helptext`:

  ```
  sp_helptext pred1
  # Lines of Text
  ---------------
             1
  text
  ------------------------------------------------------
  grant select on tab1 where col1 = 5 as pred1 to robert
  ```

  `sp_helptext` only displays results not previously hidden by `sp_hidetext`.
  This example displays the predicate with the source text hidden by `sp_hidetext`:

  ```
  Msg 18406, Level 16, State 1:
  Procedure 'sp_helptext', Line 313:
  Source text for compiled object pred1 (id = 592002109) is hidden.
  ```

- `sp_helprotect` lists the name of the predicated privilege, if any, in the predicate column.

See the *Reference Manual: Procedures*.

8.10 Chain-of-Ownership Effect on Predicated Privileges

A chain of ownership exists between a stored procedure, SQL function, view, or trigger and a table accessed by that object if the same user owns the calling object and the table.

With this chain of ownership, the owner of a stored procedure, view, SQL function, or trigger gives another user implicit access on the dependent object by granting explicit access to the user on the procedure, view, and so on. Granting access to the dependent object is unnecessary.

If a user has implicit permission on a table through a chain of ownership, the predicates are not applied from any row-filtering `grant` commands for this access.

For example, the owner of the employee table grants user "Priscilla" access to see the employee data in the payroll department:

```
grant select on employee
  where dept = 'payroll'
  to priscilla
```

When Priscilla enters:

```
select name, phone from employee
  where city='SFO'
```

The names and phone numbers are returned of only those employees who live in San Francisco and work in the payroll department.

However, if the owner of the `employee` table creates the following procedure and grants `execute` access to priscilla, the ownership chain between `employee` and `employee_addresses` is detected:

```
create procedure employee_addresses as
select name, phone from employee
```

Priscilla is given implicit access to all rows in `employee` in the context of the procedure without restrictions from the predicate on the previous `grant`.

8.11 ansi_permissions and Predicated Privileges

If an `update` or a `delete` command includes a `where` clause that references columns in a table that has rows being updated or deleted, SAP ASE checks for `select permission` privileges on these columns only if `ansi_permissions` is enabled for the session.

SAP ASE applies predicated privileges on the `update` and `delete` statements according to whether:

- `ansi_permissions` is enabled – predicated privileges from grant select are applied in addition to any predicate specified in the grant `update` statement if the table owner granted predicated `select` access

on the columns in the `where` clause. Verify that applying both predicates does not unexpectedly restrict the row set.

- `ansi_permissions` is disabled – `select` permission is not required on the columns of the updated or deleted table referenced in the `where` clause. Any predicates granted for `select` access on the table are not applied to the `update` or `delete` statement.

This example illustrates how `ansi_permissions` affects predicated privileges:

- The first `grant` allows user "Bob" to update employee salaries in the sales department, while the second `grant` allows Bob to select the employee ID and address of only those employees who are his direct reports:

```
grant update on employee (salary)
  where edept = "sales"
  to bob
grant select on employee (eid, eaddress)
  where user_name(mgrid) = USER
  to bob
```

  If Bob attempts to update an employee's salary based on `eid`, one or both of the predicates from `grant` commands in the examples above are applied:

```
update employee
  set salary = salary * 0.1
  where eid = 1006
```

- If the session has `set ansi_permissions` enabled, SAP ASE executes the `update` command after adding the predicates for both the `update` and the `select` statements:

```
update employee
  set salary = salary * 0.1
  where eid = 1006
  and edept = "sales"
  and user_name(mgrid) = USER
```

  Since Bob's predicated `select` permission allows him to search IDs only of employees who are his direct reports, Bob's predicated `select` access does not restrict the rows he can update or delete.
  If `ansi_permissions` are disabled, SAP ASE executes the `update` command as follows:

```
update employee
  set salary = salary * 0.1
  where eid = 1006
  and edept = 'sales'
```

  To summarize, if `ansi_permissions` is:
  - Disabled – Bob can update and delete rows that he is not allowed to select.
  - Enabled – the `where` clause on a predicated grant for `select` access qualifies the row set that may be updated or deleted.

> **i Note**
>
> Either one or both of the predicates from the `grant` commands in the examples above are applied.

8.12 Permissions on Accesses Made by Predicates

The grantor of a predicated privilege must have authorization to access any tables build-in or SQL functions referenced by the predicate.

For row-filtering predicates, SAP ASE verifies accesses made by the predicate at the same point in execution that is verifies accesses made by the user's `select`, `update` or `delete` commands. Accesses made by role activation predicates are checked when the user activates a role. The user who enters the command to which a predicate is attached is not required to have authorization on the predicate's referenced objects.

For example, enforcing data privacy requires you specify predicates that access tables and columns that contain the data owner's privacy policies. Access to these privacy policies should be restricted to a small set of users; data consumers should not necessarily have the right to view the internal data used to enforce policies.

SAP ASE enforces permissions on accesses made by predicated privileges against the grantor of the privilege, taking into account the grantor's direct and indirect permissions through group and default role membership. A user's default roles include roles granted through login profiles for automatic activation.

You may want to create an application security role that has permission on privacy metadata tables. You can then assign object owners this role as a default or automatically activated role so that the grant predicates have access to the necessary data.

In this example Bob, the owner of the `purchases` table, grants permission to market analysts with the `market_role` to view product purchasing information for those customers who opted to share their data (customer preferences are stored in the `privacy_db`).

```
grant select on purchases p
  where exists (select 1 from privacy_db..choices c   where p.custid = c.id)
  to market_role
```

When user Alice, who has activated the `market_role`, selects from the `purchases` table, these access checks are performed:

- Alice's direct or indirect `select` permission on `purchases`
- Bob's direct or indirect permission to select from the `privacy_db..choices` table

> **i** Note
>
> The ownership of a predicated privilege is modified when the object to which the predicated `grant` applies changes ownership by the `alter .. modify owner` command. If the new owner has not been granted permission on the objects accessed by the predicate, application of the predicate causes a run time error.

8.13 Predicated Privileges and Triggers

SAP ASE does not apply row-filtering predicates to pseudo tables. The row-filtering predicates applied to the base table restricts the data in `inserted` and `deleted` tables.

Accesses made by a trigger against regular tables are subject to row-filtering through predicates.

8.14 Recompile Predicated Privileges

If a stored procedure, trigger, or SQL function accesses tables that are controlled by predicated privileges, SAP ASE may need to recompile the procedural object for access by different users, or by the same user with one or more roles activated when the active roles differ from those activated when the plan was last compiled.

To prevent constant recompiling for different users executing the same procedure, SAP ASE attempts to select a cached plan that has the right "profile" for the current user. For example, if a procedure is initially compiled for execution by Joe with `role1` and `role2` active, the same procedure is likely recompiled for Bob, who has `role2` and `role3` active, assuming that accesses by the procedure are controlled by separate predicated grants to `role1`, `role2`, and `role3`.

If the procedure cache contains plans compiled for Bob and Joe, a copy of the plan that is less likely to require compiling is chosen.

The initial choice of a plan that uses predicates from the cache depends on matching the protection "profile" of the user with that of the plan, which consists of:

- The user ID for whom the plan was compiled. The user ID is saved in the plan's protection profile only if one or more predicated privileges on tables accessed by the procedure are granted directly to the user.
- The group ID if one or more predicated privileges are granted to a group.
- A list of role IDs if one or more predicated privileges are granted to a role.

Choosing a plan that matches the user's profile does not guarantee the plan is up to date with the required predicates for the current user's accesses. SAP ASE verifies that each `select`, `update`, and `delete` command in the procedure has been compiled with `where` clauses that reflect the current user's predicated grants.

Otherwise, the procedure must be recompiled. Predicated privileges introduces protection checking as a new decision point for plan recompilation during the execution phase.

For efficient use of the procedure cache and the sharing of stored procedure plans, you should use a role-based privacy policy. Predicated access to the objects referenced by a stored procedure should be granted to a small number of roles that can be activated by the end users.

8.15 Disallow Recursive Predicate Processing

SAP ASE returns an error if the accesses made in a row-filtering predicate or a role activation predicate are controlled by predicated grants. For example, this situation results in recursive cycles of access checking, and is not allowed:

- `table1` is protected by a `grant` with `predicate1` that references `table2`.
- `table2` is protected by a `grant` with `predicate2`, that references `table1`.

Role activation predicates have similar restrictions. For example, this situation returns an error when a user attempts to `set role r1` on:

- `table1` is protected by a predicated `grant`.
- A `grant` on `r1` references `table1` in an activation predicate.

When applying a user's default roles, for example, during login, when executing set proxy, or when evaluating predicate accesses on behalf of the grantor, SAP ASE ignores any default roles that have been granted with an activation predicate.

8.16 Information Leakage Through Predicates

Predicated privilege enforcement may leak information about rows in a table to a user who is not authorized to see the data, or may violate data consistency.

For example, if employee IDs are sensitive and unique, and user Joe is entitled to process IDs only for those employees who work in the sales department, when Joe updates the ID of an employee in the sales department, Joe can get confirmation about whether a particular ID exists in the entire table by whether or not SAP ASE issues a uniqueness-violation message.

Updateable views have the same potential for leaked information. Application designers who rely on access control through predicated privileges should disallow updates of primary keys, and should restrict the users with predicated `update` and `delete` privileges on tables with referential constraints.

In the following example, the column `cust_name` is a foreign key on the table `t_orders` that is constrained by the values of `customer.name`:

```
create table t_orders (ordernum int, orders_dt date,  cust_name char(60)
references customer.name, state char(2))
```

An employee in the orders department has `select` and `update` permission on the `t_orders` table for those orders shipped to customers in the state of Iowa.

```
grant select, update on t_orders
  where state = 'IA'
  to procure_role
```

The employee in the orders department may not have access to the customer table, which lists customers of the company nation-wide. The employee wants to know whether a certain customer from New York buys from this company. Using the customer name of the New York customer, the employee enters the following command to update an order placed in Iowa:

```
update orders set cust_name = 'Ronald Crump'
  where ordernum = 345
```

If above statement does not return a foreign key constraint violation, the employee knows that Ronald Crump is a customer of the company.

9 Granular Permissions

Granular permissions are used to grant system privileges, allowing you to construct site-specific roles with privileges to match your requirements, and restrict system administrators and database owners from accessing user data.

Grantable system privileges allow you to enforce "separation of duties," which requires that, for particular sets of operations no single individual is allowed to execute all operations within the set and "least privilege," which requires that all users in an information system should be granted as few privileges as are required to do the job.

You cannot revoke or grant one privilege from—or to—another privilege. However, privileges may overlap what the grantee can do. Possessing one privilege may imply possessing another, more granular, privilege.

Enabling granular permissions reconstructs system-defined roles (sa_role, sso_role, oper_role, and replication_role) as privilege containers consisting of a set of explicitly granted privileges. You can revoke explicitly granted system privileges from system-defined roles and regrant them to the roles.

9.1 Configuring SAP ASE to Use Granular Permissions

Configure SAP ASE to use granular permissions.

1. Enable the security and directory services license, ASE_SECDIRS.
2. Enable the `enable granular permissions` configuration parameter, which requires the sso_role, and checks out the ASE_SECDIRS license. Disabling `enable granular permissions` requires the granular system privilege, `manage security configuration` and checks the license back in.

When `enable granular permissions` is turned on, SAP ASE verifies that at least one unlocked account has the `manage server permissions` privilege, and at least one unlocked account has the `manage security permissions` privilege. When disabling `enable granular permissions`, SAP ASE verifies the server has at least one unlocked account with the sso_role, and that at least one unlocked account has the sa_role.

9.2 System Privileges

Granular permissions define server-wide and database-wide privileges.

You must grant or revoke server-wide privileges in the master database. The grantees must be roles, users, or groups in the master database. SAP ASE stores the permission information for server-wide privileges in `master.dbo.sysprotects`.

You must grant or revoke database-wide privileges in the database for which the command requiring the privilege is intended. The grantees can be users, groups, or roles in the database. SAP ASE stores the permission information for database-wide privileges in `<database_name>.dbo.sysprotects`.

Use the `grant` or `revoke` commands to grant or revoke server- and database-wide privileges. For example, to allow user Joe to dump any database, a user with the proper privilege issues this command from the master database:

```
grant dump any database to joe
```

To allow Joe to create any object on behalf of himself and on behalf of other users in database `db2`, a user with the proper privilege issues this command from `db2`:

```
grant create any object to joe
```

9.3 Effect of Privileges as Part of System-Defined Roles

The effects of granting and revoking privilege as part of system-defined roles and the `enable granular permissions` parameter.

When granting and revoking privileges included as part of system-defined roles:

- Any privileges explicitly granted to a system defined role may be revoked from the role. Once a privilege is revoked, the role holder can no longer perform operations related to the privilege on the server.
- `enable granular permissions` configuration parameter should be enabled for explicitly granted privileges to take effect. If `enable granular permissions` is disabled, system defined roles have the same privileges as in SAP ASE versions earlier than 15.7 ESD #2. These privileges are implicitly vested in the roles and cannot be revoked.

Use `sp_restore_system_role` to restore a modified system-defined role to its default privilege configuration.

9.4 Permission Management

All system privileges are managed by users with `manage server permissions`, `manage security permissions`, or `manage database permissions` privileges. Object permissions are managed by the object owners or users with `manage any object permission` privilege.

9.4.1 manage security permissions Privilege

Users with `manage security permissions` privileges can grant or revoke security-related sever-wide privileges and security-related database-wide privileges.

Table 15: Server-Wide Privileges Managed by manage security permissions privilege

Server-Wide privileges
`change password`
`checkpoint` (on `sybsecurity`)
`dump database` (on `sybsecurity`)
`load database` (on `sybsecurity`)
`manage any login`
`manage any login profile`
`manage any remote login`
`manage auditing`
manage roles
`manage security configuration`
`manage security permissions`
`online database` (on `sybsecurity`)
`own database` (on `sybsecurity`)
`set proxy`
`use database` (on `sybsecurity`)

Table 16: Database-Wide Privileges Managed by manage security permissions privilege

Database-Wide Privileges
`create encryption key`
`decrypt any table`
`manage any encryption key`
`manage column encryption key`
`manage database permissions`
`manage master key`
manage service key
`update any security catalog`

`manage security permissions` is initially explicitly granted to the sso_role on a newly installed server, and, by default, the sa account has `manage security permissions` privilege. Once you revoke `manage server permissions` from the sso_role, a user with this role cannot grant or revoke any security-related privilege.

To avoid having a user unintentionally causing the server to be locked, SAP ASE ensures the server contains at least one unlocked user account with `manage security permissions` privilege.

9.4.2 manage server permissions Privilege

Users with manage server permissions privilege can grant and revoke server-wide privilege.

Table 17: System Privileges Managed by manage server permissions privilege

`allow exceptional login`
`checkpoint any database`
`checkpoint` (on any database except `sybsecurity`)
`connect`
`create database`
`dbcc checkallock any database`
`dbcc checkcatalog any database`
`dbcc checkdb any database`
`dbcc checkindex any database`

dbcc checkstorage any database
dbcc checktable any database
dbcc checkverify any database
dbcc fix_text any database
dbcc indexalloc any database
dbcc reindex any database
dbcc tablealloc any database
dbcc textalloc any database
dbcc tune
dump any database
dump database (on any database except sybsecurity)
kill
kill any process
load any database
load database (on any database except sybsecurity)
manage any database
manage any ESP
manage any thread pool
manage cluster
manage data cache
manage disk
manage dump configurations
manage lock promotion threshold
manage resource limit
manage server
manage server configuration

`manage server permissions`
`mange execution classes`
`map external file`
`monitor server replication`
`mount any database`
online any database
`online database` (on any database except `sybsecurity`)
`own any database`
`own database` (on any database except `sybsecurity`)
`quiesce any database`
`set switch`
`set tracing`
`set tracing any process`
`show switch`
`shutdown`
`unmount any database`
`use any database`
`use database` (on any database except `sybsecurity`)

`manage server permissions` is initially explicitly granted to the sa_role on a newly installed server. Once you revoke `manage server permissions` from the sa_role, a user with sa_role cannot grant or revoke any server-wide privilege.

To avoid a user unintentionally causing the server to be locked, SAP ASE ensures the server contains at least one unlocked user account with `manage server permissions` privileges.

9.4.3 manage database permissions Privilege

Users with `manage database permissions` privilege can grant or revoke database-wide privileges except encryption key related privileges, which are managed by users with the manage security permissions privilege.

Users with the `manage database permissions` privilege cannot grant or revoke the `manage database permissions` privilege to or from other users, including themselves.

When a database is created, the `manage database permissions` privilege is not initially granted to any role or user, including the Database Owner. A user with `manage security permissions` privilege must explicitly grant `manage database permissions` privilege to a database user before that user can grant or revoke database permissions.

Table 18: Privileges Managed by manage database permissions privilege

`alter any object owner`
`alter any table`
`create any default`
`create any function`
`create any index`
`create any object`
`create any procedure`
`create any rule`
`create any table`
`create any trigger`
`create any view`
`create default`
`create function`
`create procedure`
`create rule`
`create table`
`create trigger`
`create view`
`dbcc checkallock`
`dbcc checkcatalog`
`dbcc checkdb`
`dbcc checkindex`
`dbcc checkstorage`

dbcc checktable
dbcc checkverify
dbcc fix_text
dbcc indexalloc
dbcc reindex
dbcc tablealloc
dbcc textalloc
delete any table
drop any default
drop any function
drop any object
drop any procedure
drop any rule
drop any table
drop any trigger
drop any view
execute any function
execute any procedure
identity_insert any table
identity_update any table
insert any table
manage abstract plans
manage any object permission
manage any statistics
manage any user
manage checkstorage

`manage database`
`manage replication`
`references any table`
`reorg any table`
`report checkstorage`
`select any audit table` (available only in `sybsecurity`)
`select any system catalog`
`select any table`
`setuser`
`transfer any table`
`truncate any table`
`truncate any audit table` (available only in `sybsecurity`)
`update any table`

9.4.4 manage any object permission Privileges

Users with manage any object permission privilege can grant or revoke object-specific permissions for any object owned by any database user. An object owner can grant all privileges on objects he or she owns.

Table 19: Object Permissions Managed by manage any object permission privilege

`delete`
`delete statistics`
`execute`
`identity_insert`
`identity_update`
`insert`
`references`
`select`

transfer table
truncate table
update
update statistics

9.5 Privileges Granted to System-Defined Roles

System-defined roles are granted privileges by default.

Table 20: Privileges Granted to sa_role by Default

allow exceptional login
checkpoint any database
connect
create database
dbcc checkalloc any database
dbcc checkcatalog any database
dbcc checkdb any database
dbcc checkindex any database
dbcc checkstorage any database
dbcc checktable any database
dbcc checkverify any database
dbcc fix_text any database
dbcc indexalloc any database
dbcc reindex any database
dbcc tablealloc any database
dbcc textalloc any database
dbcc tune

dump any database
kill any process
load any database
manage any database
manage any ESP
manage any execution class
manage any thread pool
manage cluster
manage data cache
manage disk
manage dump configuration
manage lock promotion threshold
manage resource limit
manage server
manage server configuration
manage server permissions
map external file
mount any database
online any database
own any database
quiesce any database
select on get_appcontext
select on list_appcontext
select on rm_appcontext
select on set_appcontext
set switch

`set tracing any process`
`show switch`
`shutdown`
`unmount any database`

Table 21: Privileges Granted to sso_role by Default

`alter any object owner` (in any database)
`change password`
`decrypt any table` (in any database)
`manage any encryption key` (in any database)
`manage any login`
`manage any login profile`
`manage any remote login`
`manage any user` (in any database)
`manage auditing`
`manage roles`
`manage security configuration`
`manage security permissions`
`select on authmech`
`show switch`
`set tracing any process`
`update any security catalog` (in any database)

Table 22: Privileges Granted to oper_role by Default

`checkpoint any database`
`dump any database`
`load any database`
`manage dump configuration`

`online any database`
`use any database`

Table 23: Privileges Ganted to replication_role by Default

`checkpoint any database`
`dump any database`
`load any database`
`manage replication` (in any database)
`monitor server replication`
`online any database`
`quiesce any database`
`truncate any table` (in any database)
`truncate any audit table` (in `sybsecurity`)

Table 24: Privileges Granted to keycustodian_role by Default

`manage any encryption key`

Use `sp_restore_system_role` (which requires `manage security permissions` privileges) to restore a role or database owner to the default role privilege configuration.

9.6 Privileges Assigned to the Database Owner

Database owners are granted privileges by default.

Table 25: Privileges Granted to Database Owners by Default

`alter any object owner`
`create default`
`create function`
`create procedure`
`create rule`
`create table`

`create trigger`
`create view`
`dbcc checkalloc`
`dbcc checkcatalog`
`dbcc checkdb`
`dbcc checkindex`
`dbcc checkstorage`
`dbcc checktable`
`dbcc checkverify`
`dbcc fix_text`
`dbcc indexalloc`
`dbcc reindex`
`dbcc tablealloc`
`dbcc textalloc`
`manage abstract plans`
`manage any user`
`manage checkstorage`
`manage database`
`manage database permissions` (`sybsecurity` only)
`manage replication`
`report checkstorage`
`select any audit table` (`sybsecurity` only)
`setuser`
`truncate any audit table` (`sybsecurity` only)

These rules apply to the database owner:

- By default, no newly added database-wide privileges, other than the ones listed above, are granted to the database owner. You must use an explicit `grant` command to grant any additional privilege to the database owner.
- By default, `setuser` privilege is explicitly granted to the database owner. To prevent the database owner from impersonating other users, revoke the `setuser` privilege from the database owner.
- Any user with `own any database` privilege or `own database` privilege on a database logs in to the database as the database owner, regardless if the user is a valid user of the database. Any object created by this user has `UID=1` in `sysobjects.uid` and their login name in `sysobjects.loginame`. If both `own any database`and `own database` privileges are revoked from this user, he or she enters the database with his or her own user ID or as a guest if he or she has not been added as a user in the database.

9.7 Granular Permissions Adds the sa_serverprivs_role

The sa_serverprivs_role is a user-defined role granted to the sa by default, and ensures the system administrator possesses all privileges necessary to run SAP ASE when `enable granular permissions` is enabled.

As a user-defined role, when granted to a login, sa_serverprivs_role is not activated automatically for the login during login by default.

You can use `alter login` to enable automatic activation of this role during login.

> **i** Note
>
> SAP recommends that you do not use sa_serverprivs_role as a regular user-defined role.

When you enable `enable granular permissions`, the sa login has the same privileges in a database as the database owner. The sa login can obtain other database-wide privileges by adding himself as a user of the database, and granting himself those privileges.

Table 26: Privileges Granted to sa_serverprivs_role by Default

`allow exceptional login`
`change password`
`checkpoint any database`
`connect`
`create database`
`dbcc checkalloc any database`
`dbcc checkcatalog any database`
`dbcc checkdb any database`

dbcc checkindex any database
dbcc checkstorage any database
dbcc checktable any database
dbcc checkverify any database
dbcc fix_text any database
dbcc indexalloc any database
dbcc reindex any database
dbcc tablealloc any database
dbcc textalloc any database
dbcc tune
dump any database
kill any process
load any database
manage any database
manage any ESP
manage any execution class
manage any login
manage any login profile
manage any remote login
manage auditing
manage cluster
manage data cache
manage disk
manage dump configuration
manage lock promotion threshold
manage resource limit

manage roles
manage security configuration
manage security permissions
manage server
manage server configuration
manage server permissions
map external file
mount any database
online any database
own any database
quiesce any database
select on authmech
select on get_appcontext
select on list_appcontext
select on rm_appcontext
select on set_appcontext
set switch
set tracing any process
show switch
shutdown
unmount any database

9.8 Default Roles Granted to the System Administrator

By default, the login sa is granted these roles.

sa_role

sso_role
oper_role
mon_role
sybase_ts_role
sa_serverprivs_role

9.9 Limiting the Power of the System Administrator and Database Owner

Any granular permissions granted to a system defined role can be revoked from the role to limit the power of that role. See examples below for some typical use cases.

Example 1:

Users with the `manage server permissions` privilege can restrict users with the sa_role from accessing user databases by revoking the `own any database` and `manage server permissions` privileges:

```
use master
revoke own any database from sa_role
revoke manage server permissions from sa_role
```

Example 2:

By default, `setuser` privilege is granted to the database owner, which enables the database owner access other users' data by impersonating that user.

Revoke the `setuser` privilege from the database owners to restrict them from accessing other users' data. To prevent database owners from granting setuser privileges to themselves, make sure that the `manage database permissions` privilege is not granted to the database owners in the databases. By default, the `manage database permissions` privilege is not granted to the database owner.

For example, to revoke `setuser` privileges from the database owner in database `db1`:

```
use db1
```

```
revoke setuser from dbo
```

Change these privileges in the `model` database to make this the default behavior in any user database created in the future:

```
use model
```

```
revoke setuser from dbo
```

Example 3:

Any sa_role user may accidentally shut down the server. To prevent this, a system administrator with `manage server permissions` privilege can revoke `shutdown` privilege from sa_role and grant it only to the administrators responsible for shutting down the server operation.

For example, to grant users joe and bob (both with the sa_role) the `shutdown` privilege, and revoke it from all others, a user with the `manage server permissions` privilege issues:

```
use master
grant shutdown to joe, bob
revoke shutdown from sa_role
```

9.10 Enable granular permissions and sybsecurity

These restrictions apply to the `sybsecurity` database when you enable `enable granular permissions`.

- Only users with `manage auditing` privilege can create `sybsecurity`.
- Any user with `manage auditing` privilege can access `sybsecurity` as the database owner.
- By default, `manage database permissions`, `select any audit table`, and `truncate any audit table` privileges are granted to the database owner of `sybsecurity`.
- All server privileges that include the word "any" do not apply to `sybsecurity`. For example, `own any database` does not grant the privilege holder access to `sybsecurity` as the database owner.
- Only users with `manage security permissions` can grant `own database`, `dump database`, `load database`, `checkpoint`, and `online database` privileges on `sybsecurity` to other users.

9.11 Logging in to a Locked-Out Server

If `enable granular permissions` is enabled, a SAP ASE server can be locked out only if all logins with the `change password` privilege lose their passwords. The `dataserver` command line includes the following parameters to unlock the server.

- `-p <login_name>` – specifies the login name when starting the server so this account's password may be reset. A random password is generated, displayed, encrypted, and saved in `master..syslogins` as a new password for this account. When granular permissions is disabled, the `<login_name>` must have sso_role. When granular permission is enabled, the `<login_name>` must have `change password` privilege.
- `-u <login_name>` – specifies a login name you want to unlock. When granular permissions is disabled, the `<login_name>` must have either sso_role or sa_role. When granular permissions is enabled, the `<login_name>` must have `change password` privilege.
- `-A system_role`, `--role-logins` – specifies the system role name so that a list of login accounts with this role is printed into log file.
- `-n system_privilege`, `--permission-logins` – specifies the system privilege name so that a list of login accounts with this system privilege is printed into log file.

To unlock a locked-out server, the account name specified with the `-u` and `-p` parameters must be a login name that has the `change password` privilege. Once that user has logged in to the server with the new password, he or she should first reset their own password to a new password then they can reset passwords for other logins.

To generate a list of login accounts that have sso_role when granular permissions is not enabled, use the `-A <system_role>` or `--role-logins` parameter. For example:

```
$SYBASE/$SYBASE_ASE/bin/dataserver -d master.dat -s server_name-A sso_role
```

To generate a list of login accounts that has `change password` privilege when granular permissions is enabled, use the `-n <system_privilege>` or `--permission-logins` parameters. For example:

```
$SYBASE/$SYBASE_ASE/bin/dataserver -d master.dat -s server_name-n "change
password"
```

9.12 General Use Scenarios

Examples of how to set permissions for general use scenarios.

Scenario 1: Permissions for an Application Server User

Example of allowing a user to execute DML commands on all user tables and execute any stored procedures and user-defined functions.

A user with `manage database permissions` privilege in database `db1` issues the following to allow user app_user to execute DML commands on all user tables and execute any stored procedures and user-defined functions defined in `db1`:

```
use db1
grant insert any table to app_user
grant select any table to app_user
grant update any table to app_user
grant truncate any table to app_user
grant delete any table to app_user
grant identity_insert any table to app_user
grant identity_update any table to app_user
grant execute any procedure to app_user
grant execute any function to app_user
```

Scenario 2: Permissions for a Database Access Manager

Example of allowing a user to manage users, groups, aliases, and permissions on all objects.

A user with the `manage database permissions` privilege in database `db2` issues the following to allow user Joe to manage users, groups, aliases, and permissions on all objects in `db2`:

```
use db2
grant manage any object permission to joe
grant manage any user to joe
```

Scenario 3: Permissions for a Database Backup Manager

Example of allowing a user to run `dump database`, `dump transaction`, `checkpoint`, and `quiesce database` on all user databases.

A user with the `manage server permissions` privilege issues the following to allow user Mike, who is a user in master database, but not in other user databases, to run `dump database`, `dump transaction`, `checkpoint`, and `quiesce database` on all user databases:

```
use master
grant dump any database to mike
grant use any database to mike
grant checkpoint any database to mike
grant quiesce any database to mike
```

Scenario 4: Permissions for a Help Desk Operator

Example of allowing a help desk operator to reset passwords or kill processes.

A user with the `manage security permissions` privilege issues the following to allow help desk operator Alice to reset passwords for users who call in with password issues:

```
use master
grant change password to alice
```

A user with the `manage server permissions` privilege issues the following to allow Alice to kill runaway user processes that are using system resources:

```
use master
grant kill any process to alice
```

Scenario 5: Permissions for a Security Auditor

Example of allowing an information security department employee to generate daily reports from the SAP ASE auditing subsystem.

A user with the `manage auditing` privilege issues the following to allow information security department employee Jane to generate daily reports from the SAP ASE auditing subsystem:

```
use sybsecurity
```

```
sp_adduser jane
grant select any audit table to jane
```

9.13 System Table master.dbo.sysprotects

When a server-wide privilege such as `own database` is granted on a database, the permission is recorded in `master.dbo.sysprotects`, with the database ID (dbid) stored in the ID field.

All grantable permission values are stored in `master.dbo.spt_values`. The columns in `spt_values` are overloaded as:

Table 27: master.dbo.spt_values

Parameter	Value
`name`	Permission name string.
`number`	Internal token number of the permission (`sysprotects.action`).
`type`	Identifier for permission entries:
	`T` – permissions available and grantable in releases prior to 15.7 ESD #1 and earlier and continue to be available and grantable in 15.7 ESD #2 and later whether the granular permissions option is on or off.
	`GP` – permissions only available and grantable when granular permissions option is on.
`ansi_w`	Execution scope of the permission:
	1 – server-wide
	2 – database-wide
	4 – object
`low`	SAP ASE version in which the permission is first introduced:
	0 – exist in versions earlier than 15.7 ESD #2.
	157002 – introduced in 15.7 ESD #2.
`high`	Token number of the permission that manages the current permission when granular permissions option is on:
	117 – `manage server permissions`
	114 – `manage security permissions`
	106 – `manage database permissions`

Parameter	Value
	97 – `manage any object permission`
`msgnum`	Bitset of the permission token: • `0x00000001 server-wide privilege` • `0x00000002 database-wide privilege` • `0x00000004 object permission` • `0x00000008 when granting the privilege a database name is required` • `0x00000010 the privilege is managed by "manage server permissions"` • `0x00000020 the privilege is managed by "manage security permissions"` • `0x00000040 the privilege is managed by "manage database permissions"` • `0x00000080 column level permission` • `0x00000100 the permission applies to system table object` • `0x00000200 the permission applies to view object` • `0x00000400 the permission applies to user table object` • `0x00000800 the permission applies to procedure object`

9.14 Database User usedb_user Account

By default, the database user account `usedb_user` is added to each database. When granular permissions is enabled, a user accessing a database will assume the user name `usedb_user` if the user is granted the `use database` privilege on the database.

The following conditions apply to the user:

- Does not have `own database` privilege on the database
- Does not hold an identity as a valid user in the database
- Is not aliased to another valid user in the database

The following rules apply to the user account `usedb_user`:

- `usedb_user` is only authorized to perform operations in the database that are allowed for `public`. To perform any additional operations, the user must acquire the corresponding privileges. A user may acquire privileges in a database through roles when he or she is not a valid user in the database.
- To create an object, the user must be a valid user in the database.

Allowing a user to access a database as `usedb_user` will provide the user the ability to execute a server-wide privilege in a database without being added as a valid user in the database.

For example, Bob is a valid user in `master` database, but not a valid user for database `db1` which does not have a guest user account. Bob has `manage security permissions` privileges in the `master` database.

To allow Bob to execute `manage security permissions` in `db1`, a user with `manage server permissions` privilege can issue:

```
grant use database on db1 to bob
```

Bob can now access database `db1` and issue the commands to grant `manage any encryption key` privileges to user Alice:

```
use db1
grant manage any encryption key to alice
```

SAP ASE records the grantor of `manage any encryption key` in `sysprotects` as the user ID of `usedb_user`.

9.15 Grantable System Privileges

These are the grantable server-wide and database-wide privileges.

Privileges marked with an asterisk (*) may be granted or revoked when `enable granular permissions` is disabled.

For a list of all grantable privileges and permissions in alphabetic order, see *Privileges for grant* in the `grant` command section in *Reference Manual: Commands*.

> **i** Note
>
> Possessing one privilege may imply possessing another, more granular, privilege. For example, a user with `select any table` privilege implies the user has `select` permission on all user tables. See *Privileges for grant* in the `grant` command section of *Reference Manual: Commands* for a complete list of privileges pairs that have an implied relationship.

Table 28: Server-Wide Privileges

Privilege name	Operations the privilege authorizes
Privileges management	
`manage security permissions`	Granting and revoking security privileges
`manage server permissions`	Granting and revoking nonsecurity server-wide privileges not related to security.
Audit management	

Privilege name	Operations the privilege authorizes
`manage auditing`	Acting as the database owner of `sybsecurity` Executing these commands: • `create database sybsecurity` • `alter database` sybsecurity • `drop database sybsecurity` • `set switch(7601)` • `truncate table` (audit table) Executing these system store procedures: • `sp_addaudittable` • `sp_audit` • `sp_displayaudit`
Login and role management	
`allow exceptional login`	Granting login to the server when: • Server maximum connection limit exceeded • Master database is in restore mode • Server is in shutdown • Recovery is in progress (during server restart)
`change password`	Executing this command: • `alter login ... change password` Executing these system store procedure: `sp_locklogin` (unlock any login account which was locked because the user exceeded the limit of maximum failed logins)

Privilege name	Operations the privilege authorizes
`manage any login`	Executing these commands: • `create login` • `alter login` • `drop login` Executing these system procedures: • `sp_autoconnect` • `sp_defaultdb` • `sp_displaylogin` • `sp_grantlogin` (Windows only) • `sp_helpmaplogin` • `sp_logininfo` (Windows only) • `sp_locklogin` • `sp_maplogin` • `sp_revokelogin` (Windows only) Executing this function: • `valid_user`
`manage any login profile`	Executing these commands: • `create login profile` • `alter login profile` • `drop login profile` Executing this system procedure: • `sp_securityprofile`
`manage any remote login`	Executing these system procedures: • `sp_addexternlogin` • `sp_addremotelogin` • `sp_dropexternlogin` • `sp_dropremotelogin` • `sp_dropserver` • `sp_remoteoption`

Privilege name	Operations the privilege authorizes
`manage roles`	Executing these commands: • `create role` • `alter role` • `drop role` • `grant role` • `revoke role` Executing these system procedures: • `sp_displayroles` • `sp_grantlogin` • `sp_logininfo` • `sp_revokelogin`
Database Management	
`checkpoint`	Executing the `checkpoint` command for a specified database
`checkpoint any database`	Executing the `checkpoint` command for any database
`create database*`	Executing the `create database` command
`dump any database`	Executing these commands for any database: • `dump database` • `dump transaction`
`dump database`	Executing these commands for a specified database: • `dump database` • `dump transaction`
`load any database`	Executing these commands for any database: • `load database` • `load transaction`
`load database`	Executing these commands for a specified database: • `load database` • `load transaction`

Privilege name	Operations the privilege authorizes
`manage any database`	Performing database maintenance operations on any database Executing these commands: • `install jar` • `remove jar class` Executing these system procedures: • `sp_addobjectdef` • `sp_addsegment` • `sp_addthreshold` • `sp_checksource` • `sp_dbextend 'simulate'` • `sp_dbextend 'execute'` • `sp_dbextend 'clear','threshold'` • `sp_dbextend 'check'` • `sp_dbextend 'set','threshold'` • `sp_dbextend 'modify', 'database'` • `sp_dbextend 'set','database'` • `sp_dbextend 'set', 'threshold'` • `sp_dropobjectdef` • `sp_dropsegment` • `sp_dropthreshold` • `sp_droptype` • `sp_extendsegment` • `sp_hidetext` • `sp_modifthreshold` • `sp_placeobject` • `sp_procxmode` • `sp_rename` • `sp_rebuild_text` • `sp_spaceusage` (for some parameters)

Privilege name	Operations the privilege authorizes
`manage any database` (continued)	Executing these `dbcc` commands: • `dbcc dbrepair(remap)` • `dbcc dbrepair(newthreshold)` • `dbcc dbrepair(findstranded)` • `dbcc dbrepair(fixlogfreespace)` • `dbr_remap` • `dbcc dbrepair(ltmignore)` • `dbcc dbrepair(upd_usg)` • `dbcc dbrepair(auinit)` • `dbcc dbrepair(dmap_unlock)` • `dbcc rebuild_text` • `dbcc refreshids (placeobject)` • `dbcc refreshpdes` • `dbcc update_tmode` • `dbcc upgrade_obj` Executing these functions: • `derived_stat` • `identity_burn_max`
`mount any database`	Executing the `mount database` command for any database
`online any database`	Executing the `online database` command for any database
`online database`	Executing the `online database` command for a specified database
`own any database`	Acting as database owner for the specified database See `own database` for a list of operations the privilege is authorized to perform.

Privilege name	Operations the privilege authorizes
`own database`	Acting as database owner for any database except `sybsecurity` Executing these commands: • `alter database` • drop database • `grant` (default) • `revoke` (default) • `checkpoint` • `dump database` • `dump transaction` • `load database` • `load transaction` • `online database` • `use database`
`own database` (continued)	Executing these system procedures: • `sp_addmessage` • `sp_altermessage` • `sp_dboption` • `sp_dbremap` • `sp_dropmessage` • `sp_fixindex` • `sp_forceonline_db` • `sp_forceonline_page` • `sp_helptext` • `sp_logdevice` • `sp_post_xpload` • `sp_renamedb` • `sp_setsuspect_granularity` • `sp_tempdb_markdrop` • `sp_version` • `xp_cmdshell` • `xp_enumgroups` • `xp_logevent` Executing these dbcc commands: • `dbcc addtempdb` • `dbcc dbrepair` • `dbcc reindex`

Privilege name	Operations the privilege authorizes
`quiesce any database`	Executing the `quiesce database` command for any database
`unmount any database`	Executing the `unmount database` command for any database
Server Maintenance	
`manage any thread pool`	Executing these commands: • `create thread pool` • `alter thread pool` • `drop thread pool`
`manage cluster`	Managing cluster-related configurations and operations Executing the `shutdown cluster` command (requires `shutdown` privilege) Executing these system procedures: • `sp_addserver` • `sp_cluster` • `sp_clusterlockusage` • `sp_dropserver` • `sp_serveroption` • `sp_tempdb_markdrop` Executing these `dbcc` commands: • `dbcc quorum` • `dbcc set_scope_in_cluster`
`manage disk`	Executing these commands: • `disk init` • `disk refit` • `disk reinit` • `disk mirror` • `disk unmirror` • `disk remirror` Executing these system procedures: • `sp_addumpdevice` • `sp_diskdefault` • `sp_deviceattr` • `sp_dropdevice` • `sp_refit_admin` • `sp_dbextend` (required by some options)

Privilege name	Operations the privilege authorizes
`manage security configuration`	Enable or disable security related configurations Executing these system procedures: • `sp_configure` (to set security-related configuration options) • `sp_encryption` • `sp_logintrigger` • `sp_ldapadmin` • `sp_passwordpolicy` • `sp_ssladmin`

Privilege name	Operations the privilege authorizes
`manage server`	Manage server maintenance operations Executing these system procedures: • `sp_addlanguage` • `sp_addserver` (current security system officer) • `sp_clearstats` • `sp_countmetadata` • `sp_dbrecovery_order` • `sp_displaylogin` • `sp_displayroles` • `sp_droplanguage` • `sp_dropserver` (current security system officer) • `sp_engine` • `sp_errorlog` • `sp_extengine` • `sp_helpapptrace` • `sp_metrics` • `sp_monitorconfig` • `sp_object_stats` • `sp_reportstats` • `sp_serveroption` • `sp_setlangalias` • `sp_tempdb` Executing these `dbcc` commands: • `dbcc complete_xact` • `dbcc engine` • `dbcc forget_xact` • `dbcc traceflags` Executing these functions: • `passinfo` • `valid_user`

Privilege name	Operations the privilege authorizes
`manage server configuration`	Enable or disable server configurations not related to security Executing these system procedures: • `sp_configure` (set security related configuration options) • `sp_displaylevel` • `sp_jreconfig` • `sp_lmconfig` • `sp_pciconfig`
`shutdown`	Shutting down the: • Server • Cluster (also requires `manage cluster` privilege) • Instance • Backup Server Execute the `shutdown` command.
dbcc	
`dbcc checkalloc any database`	Executing `dbcc checkalloc` in any database
`dbcc checkcatalog any database`	Executing `dbcc checkcatalog` in any database
`dbcc checkdb any database`	Executing `dbcc checkdb` in any database
`dbcc check index any database`	Executing `dbcc checkindex` in any database
`dbcc checkstorage any database`	Executing `dbcc checkstorage` in any database
`dbcc checktable any database`	Executing `dbcc checktable` in any database
`dbcc checkverify any database`	Executing `dbcc checkverify` in any database
`dbcc fix_text any database`	Executing `dbcc fix_text` in any database
`dbcc indexalloc any database`	Executing `dbcc indexalloc` in any database
`dbcc reindex any database`	Executing `dbcc reindex` in any database
`dbcc tablealloc any database`	Executing `dbcc tablealloc` in any database
`dbcc textalloc any database`	Executing `dbcc textalloc` in any database
`dbcc tune`	Executing `dbcc tune`
Application Management	

Privilege name	Operations the privilege authorizes
`manage any execution class`	Executing these system procedures: • `sp_addengine` • `sp_addexeclass` • `sp_bindexeclass` • `sp_clearpsexe` • `sp_dropengine` • `sp_dropexeclass` • `sp_setpsexe` • `sp_unbindexeclass`
`manage any ESP`	Executing these system procedures: • `sp_addextendedproc` • `sp_dropextendedproc` • `sp_freedll` • `sp_helpextendedproc`
`manage data cache`	Executing these system procedures: • `sp_bindcache` • `sp_cacheconfig` • `sp_cachestrategy` • `sp_logiosize` • `sp_poolconfig` • `sp_unbindcache` • `sp_unbindcache_all`
manage dump configuration	Managing dump configuration for a backup server Executing these commands: • `dump configuration` Executing these system procedures: • `sp_config dump` • `sp_dump history`
`manage lock promotion threshold`	Executing these system procedures: • `sp_dropglockpromote` • `sp_droprowlockpromote` • `sp_setpglockpromote` • `sp_setrowlockpromote`

Privilege name	Operations the privilege authorizes
`monitor qp performance`	Monitoring query processing performance Executing these commands: • `set switch (3604, 3605)` • `set tracefile` • `set plan for plan_list` • `set option <><optimizer_show_option> {value | on |off}` Executing these system procedures: • `sp_cmp_all_qplans` • `sp_cmp_qplans` • `sp_find_qplans` • `sp_flush_query_tuning` • `sp_flushmetrics` • `sp_flushstats` • `sp_metrics (for 'filter', 'show', 'help')` • `sp_showplan` Executing these `dbcc` commands: • `dbcc traceoff(3604, 3605)` • `dbcc traceon(3604, 3605)` • `dbcc nodetraceoff(3604, 3605)` • `dbcc nodetraceon(3604, 3605)`
`manage resource limit`	Executing these system procedures: • `sp_add_resource_limit` • `sp_add_time_range` • `sp_drop_resource_limit` • `sp_drop_time_range` • `sp_help_resource_limit` • `sp_modify_resource_limit` • `sp_modify_time_range`
Others	
`connect*`	Connecting to any server using the `connect` command
`kill`	Killing processes owned by the privilege holder
`kill any process`	Killing any process owned by any user
map external file	Mapping a proxy table to a directory or file on a remote server

Privilege name	Operations the privilege authorizes
`monitor server replication`	Displaying replication status Executing these system procedures: • `sp_config_rep_agent` (no configure value specified, with or without database name specified) • `sp_help_rep_agent` (with or without the database name specified)
`set proxy`	Executing `set proxy` to change the identity to another user
`set tracing*`	Executing these commands: • `set tracefile` (for your own session) • `set plan for <plan_list> on|off` • `set option <><optimizer_show_option>on|off` Executing these `dbcc` commands: • `dbcc traceoff(3604, 3605)` • `dbcc traceon(3604, 3605)` • `dbcc nodetraceoff(3604, 3605)` • `dbcc nodetraceon(3604, 3605)`
`set tracing any process`	Executing these commands: • `set tracefile` (for any session) • `set plan for <plan_list> on|off` • `set option <><optimizer_show_option> on|off` Executing these `dbcc` commands: • `dbcc traceoff(3604, 3605)` • `dbcc traceon(3604, 3605)` • `dbcc nodetraceoff(3604, 3605)` • `dbcc nodetraceon(3604, 3605)`

Privilege name	Operations the privilege authorizes
`set switch`	Enabling or disabling any trace flag Executing these commands: • `set switch` • `show switch` Executing these `dbcc` commands: • `dbcc traceon` • `dbcc traceoff` • `dbcc nodetraceon` • `dbcc nodetraceoff` Executing this stored procedure: • `sp_dbextend 'trace'`
`show switch`	Displays traceflags that are on Execute the `show switch` command
`use any database`	Accessing any database when the privilege holder is not a valid user of the database and there is no “guest” account in the database Execute the `use database` command
`use database`	Accessing the specified database when the privilege holder is not a valid user of the database and there is no guest account in the database Execute the `use database` command

Table 29: Database-Wide Privileges

Privilege name	Operations this privilege authorizes
Permission management	
`manage any object permission`	Granting and revoking object permissions
`manage database permissions`	Granting and revoking database privileges
Manage user	

Privilege name	Operations this privilege authorizes
`manage any user`	Executing these system procedures: • `sp_addalias` • `sp_addgroup` • `sp_adduser` • `sp_changegroup` • `sp_dropalias` • `sp_dropgroup` • `sp_dropuser`
Set user	
`setuser`	Impersonating another user
Replication Management	
`manage replication`	Managing replication settings in a database Executing these commands: • `set replication` • `set repmode` • `set repthreshold` Executing these system procedures: • `sp_config_rep_agent` (with database name specified) • `sp_help_rep_agent` (with database name specified) • `sp_replication_path` • `sp_reptostandby` • `sp_setrepcol` • `sp_setrepdb` • `sp_setrepdbmode` • `sp_setrepdefmode` • `sp_setreplicate` • `sp_setrepproc` • `sp_setreptable` • `sp_start_rep_agent` • `sp_stop_rep_agent` Executing these `dbcc` commands: • `dbcc gettrunc` • `dbcc settrunc`
Maintains database	

Privilege name	Operations this privilege authorizes
`manage database`	Performing database maintenance operations without accessing dbo-owned data Executing these commands: • `install jar` • `remove jar class` Executing these system procedures: • `sp_addobjectdef` • `sp_addsegment` • `sp_addthreshold` • `sp_checksource` • `sp_dropobjectdef` • `sp_dropsegment` • `sp_dropthreshold` • `sp_droptype` • `sp_extendsegment` • `sp_hidetext` • `sp_merge_dup_inline_default` • `sp_modifthreshold` • `sp_placeobject` • `sp_procxmode` • `sp_rebuild_text` • `sp_spaceusage` (for some parameters)

Privilege name	Operations this privilege authorizes
`manage database` (continued)	Executing these `dbcc` commands: • `dbcc dbrepair(remap)` • `dbcc dbrepair(newthreshold)` • `dbcc dbrepair(findstranded)` • `dbcc dbrepair(fixlogfreespace)` • `dbcc dbrepair(ltmignor)` • `dbcc dbrepair(updusg_anchors)` • `dbcc dbrepair(auinit)` • `dbcc dbrepair(dmap_unlock)` • `dbcc rebuild_text` • `dbcc refreshids (placeobject)` • `dbcc refreshpdes` • `dbcc update_tmode` • `dbcc upgrade_obj` Executing these built-in functions: • `derived_stat` • `identity_burn_max` • lct_admin • `next_identity`
Manage query plan	
`manage abstract plans`	Executing these system procedures: • `sp_add_qpgroup` • `sp_cmp_all_qplans` • `sp_cmp_qplans` • `sp_copy_all_qplans` • `sp_drop_all_qplans` • `sp_drop_qpgroup` • `sp_drop_qplan` • `sp_export_qpgroup` • `sp_find_qplan` • `sp_help_qpgroup` • `sp_help_qplan` • `sp_import_qpgroup` • `sp_rename_qpgroup` • `sp_set_qplan`
dbcc	

Privilege name	Operations this privilege authorizes
`dbcc checkalloc*`	Executing `dbcc checkalloc` in the database
`dbcc checkcatalog*`	Executing `dbcc checkcatalog` in the database
`dbcc checkdb*`	Executing `dbcc checkdb` in the database
`dbcc checkindex*`	Executing `dbcc checkindex` in the database
`dbcc checkstorage*`	Executing `dbcc checkstorage` in the database
`dbcc checktable*`	Executing `dbcc checktable` in the database
`dbcc checkverify*`	Executing `dbcc checkverify` in the database
`dbcc fix_text*`	Executing `dbcc fix_text` in the database
`dbcc indexalloc*`	Executing `dbcc indexalloc` in the database
`dbcc reindex*`	Executing `dbcc reindex` in the database
`dbcc tablealloc*`	Executing `dbcc tablealloc` in the database
`dbcc textalloc*`	Executing `dbcc textalloc` in the database
`manage checkstorage`	Managing `dbcc checkstorage`-related settings on the database (specified with the procedures in which the privilege is granted) Executing these `dbcc stored` procedures: • `sp_dbcc_deletedb` • `sp_dbcc_deletehistory` • `sp_dbcc_evaluatedb` • `sp_dbcc_exclusions` • `sp_dbcc_patch_finishtime` • `sp_dbcc_updateconfig`

Privilege name	Operations this privilege authorizes
`report checkstorage`	Executing `dbcc` procedures to generate reports about `dbcc checkstorage` results on the database (specified with the procedures in which the privilege is granted) Executing these `dbcc` stored procedures: • `sp_dbcc_configreport` • `sp_dbcc_differentialreport` • `sp_dbcc_faultreport` • `sp_dbcc_fullreport` • `sp_dbcc_recommendations` • `sp_dbcc_statisticsreport` • `sp_dbcc_summaryreport`
System Catalog	
`select any audit table`	Selecting any audit table in `sybsecurity` (available only in `sybsecurity` database)
`select any system catalog`	Selecting all columns from any system table in the current database
`truncate any audit table`	Truncating any audit table in `sybsecurity` (available only in the `sybsecurity` database)
`update any security catalog`	Updating, inserting, and deleting these security-related system catalogs, which are restricted from direct update: • `master.dbo.syslogins` • `master.dbo.syssrvroles` • `master.dbo.sysloginroles` • `db.dbo.sysroles` • `db.dbo.sysprotects` **i Note** Configuration parameter `allow updates to system tables` must be enabled before any catalogs can be updated.
Manage objects	
`alter any object owner`	Altering ownership for any object in the database Execute the `alter ... modify owner` command.

Privilege name	Operations this privilege authorizes
`create any object`	Creating any of these objects owned by anyone: • tables • views • procedures • functions • defaults • rules • indexes • triggers Executing these commands: • `create table` • `create view` • `create procedure` • `create function` • `create rule` • `create default` • `create trigger` • `create index`
`drop any object`	Dropping any of these objects owned by anyone: • tables • views • procedures • functions • defaults • rules • indexes • triggers Executing these commands: • `drop default` • `drop function` • `drop index` • `drop procedure` • `drop rule` • `drop table` • `drop trigger` • `drop view`
Manage encryption	

Privilege name	Operations this privilege authorizes
`create encryption key*`	Creating encryption keys in the database
`manage any encryption key`	Creating, altering , and dropping column encryption keys, master keys, and service keys owned by anyone Executing these commands: • `create encryption key` • `alter encryption key` • `drop encryption key` Executing `sp_encryption`
`manage column encryption key`	Creating, altering, and dropping column encryption keys
`manage master key`	Creating, altering, and dropping master keys
`manage service key`	Creating, altering, and dropping service keys
Defaults	
`create default*`	Creating self-owned default Execute the `create default` command
`create any default`	Creating defaults owned by anyone Execute the `create default` command
`drop any default`	Dropping defaults owned by anyone Execute the `drop default` command
Functions	
`create function*`	Creating self-owned user-defined function Executing these commands: • `create function` • `create function (SQLJ)`
`create any function`	Creating functions owned by anyone Executing these commands: • `create function` • `create function (SQLJ)`

Privilege name	Operations this privilege authorizes
`drop any function`	Dropping functions owned by anyone Executing these commands: • `drop function` • `drop function` (SQLJ)
`execute any function`	Running user-defined functions owned by anyone Execute the `execute` command
Indexes	
`create any index`	Creating indexes on tables owned by anyone Execute the `create index` command
Procedures	
`create procedure`	Creating self-owned procedures Execute the `create procedure` command
`create any procedure`	Creating procedures owned by anyone Execute the `create procedure` command
`execute any procedure`	Execute procedures owned by anyone Execute the `execute` command.
`drop any procedure`	Dropping procedures owned by anyone Execute the `drop procedure` command
Rules	
`create rule*`	Creating self-owned rule Execute the `create rule` command
`create any rule`	Creating rule owned by anyone Execute the `create rule` command
`drop any rule`	Dropping rules owned by anyone Execute the `drop rule` command.
Tables	
`alter any table`	Altering user tables owned by anyone Execute the `alter table` command.

Privilege name	Operations this privilege authorizes
`create any table`	Creating user tables owned by anyone Execute the `create table` command.
`create table*`	Creating self-owned user tables Execute the `create table` command
`decrypt any table`	Decrypting any encrypted table
`delete any table`	Delete rows of user tables owned by anyone Executing these commands: • `delete table` • `lock table`
`drop any table`	Dropping user tables owned by anyone Execute the `drop table` command.
`identity_insert any table`	Enabling or disabling identity_update on any user table Execute the `set identity_insert` command
`identity_update any table`	Enabling or disabling identity_insert on any user table Execute the `set identity_update` command
`insert any table`	Inserting user tables owned by anyone Executing these commands: • `insert` • `lock table`
`manage any statistics`	Update or delete statistics on any table owned by anyone Executing these commands: • `delete statistics` • `update statistics` Executing `sp_modifystats`
`references any table`	Referencing user tables owned by anyone
`reorg any table`	Reorganizing user tables owned by anyone Execute the `reorg` command

Privilege name	Operations this privilege authorizes
`select any table`	Selecting user tables owned by anyone Execute these commands: • `select` • `lock table` (for share lock) • `readtext`
`transfer any table`	Transferring data to or from user tables owned by anyone Execute the `transfer table` command
`truncate any table`	Truncating user tables owned by anyone Execute the `truncate table` command
`update any table`	Updating user tables owned by anyone Execute these commands: • `update` • `lock table` • `writetext`
Trigger	
`create trigger*`	Creating self-owned trigger. Execute the `create trigger` command.
`create any trigger`	Creating triggers owned by anyone Execute the `create trigger` command
`drop any trigger`	Dropping triggers owned by anyone Execute the `drop trigger` command
Views	
`create view*`	Creating self-owned view Execute the `create view` command
`create any view`	Creating views owned by anyone Execute the `create view` command.
`drop any view`	Dropping views owned by anyone Execute the `drop view` command.

10 Confidentiality of Data

SAP ASE security services support Secure Sockets Layer (SSL) session-based security.

SSL is the standard for securing the transmission of sensitive information, such as credit card numbers, stock trades, and banking transactions, over the Internet.

While a comprehensive discussion of public-key cryptography is beyond the scope of this document, the basics are worth describing so that you have an understanding of how SSL secures Internet communication channels. This document is not a comprehensive guide to public-key cryptography.

The implementation of SSL features assume that there is a knowledgeable system security officer who is familiar with the security policies and needs of your site, and who has general understanding of SSL and public-key cryptography.

TCP/IP is the primary transport protocol used in client/server computing, and is the protocol that governs the transmission of data over the Internet. TCP/IP uses intermediate computers to transport data from sender to recipient. The intermediate computers introduce weak links to the communication system where data may be subjected to tampering, theft, eavesdropping, and impersonation.

10.1 Public-Key Cryptography

Several mechanisms, known collectively as public-key cryptography, have been developed and implemented to protect sensitive data during transmission over the Internet. Public-key cryptography consists of encryption, key exchange, digital signatures, and digital certificates.

Encryption

Encryption is a process wherein a cryptographic algorithm is used to encode information to safeguard it from anyone except the intended recipient. There are two types of keys used for encryption:

- Symmetric-key encryption – is where the same algorithm (key) is used to encrypt and decrypt the message. This form of encryption provides minimal security because the key is simple, and therefore easy to decipher. However, transfer of data that is encrypted with a symmetric key is fast because the computation required to encrypt and decrypt the message is minimal.
- Public/private key encryption – also known as asymmetric-key, is a pair of keys that are made up of public and private components to encrypt and decrypt messages. Typically, the message is encrypted by the sender with a private key, and decrypted by the recipient with the sender's public key, although this may vary. You can use a recipient's public key to encrypt a message, who then uses his private key to decrypt the message.
  The algorithms used to create public and private keys are more complex, and therefore harder to decipher. However, public/private key encryption requires more computation, sends more data over the connection, and noticeably slows data transfer.

Key Exchange

The solution for reducing computation overhead and speeding transactions without sacrificing security is to use a combination of both symmetric key and public/private key encryption in what is known as a key exchange.

For large amounts of data, a symmetric key is used to encrypt the original message. The sender then uses either his private key or the recipient's public key to encrypt the symmetric key. Both the encrypted message and the encrypted symmetric key are sent to the recipient. Depending on what key was used to encrypt the message (public or private) the recipient uses the opposite to decrypt the symmetric key. Once the key has been exchanged, the recipient uses the symmetric key to decrypt the message.

Digital Signatures

Digital signatures are used for tamper detection and non-repudiation. Digital signatures are created with a mathematical algorithm that generates a unique, fixed-length string of numbers from a text message; the result is called a hash or message digest. To ensure message integrity, the message digest is encrypted by the signer's private key, then sent to the recipient along with information about the hashing algorithm. The recipient decrypts the message with the signer's public key. This process also regenerates the original message digest. If the digests match, the message proves to be intact and tamper free. If they do not match, the data has either been modified in transit, or the data was signed by an impostor. Further, the digital signature provides non-repudiation—senders cannot deny, or repudiate, that they sent a message, because their private key encrypted the message. Obviously, if the private key has been compromised (stolen or deciphered), the digital signature is worthless for non-repudiation.

Digital Certificates

Digital Certificates are like passports: once you have been assigned one, the authorities have all your identification information in the system. Like a passport, the certificate is used to verify the identity of one entity (server, router, Web sites, and so on) to another.

SAP ASE uses two types of certificates:

- Server certificates – a server certificate authenticates the server that holds it. Certificates are issued by a trusted third-party Certificate Authority (CA). The CA validates the holder's identity, and embeds the holder's public key and other identification information into the digital certificate. Certificates also contain the digital signature of the issuing CA, verifying the integrity of the data contained therein and validating its use.
- CA certificates (also known as trusted root certificates) – is a list of trusted CAs loaded by the server at start-up. CA certificates are used by servers when they function as a client, such as during remote procedure calls (RPCs). SAP ASE loads its CA trusted root certificate at start-up. When connecting to a remote server for RPCs, SAP ASE verifies that the CA that signed the remote server's certificate is a "trusted" CA listed in its own CA trusted roots file. If it is not, the connection fails.

Certificates are valid for a period of time and can be revoked by the CA for various reasons, such as when a security breach has occurred. If a certificate is revoked during a session, the session connection continues. Subsequent attempts to login fail. Likewise, when a certificate expires, login attempts fail.

The combination of these mechanisms protect data transmitted over the Internet from eavesdropping and tampering. These mechanisms also protect users from impersonation, where one entity pretends to be another (spoofing), or where a person or an organization says it is set up for a specific purpose when the real intent is to capture private information (misrepresentation).

10.2 SAP Common Crypto Library

SAP ASE uses the proprietary SAP Common Crypto Library as the SSL provider and cryptography library. Beginning with SAP ASE version 16.0 SP03, the OpenSSL library is not supported for encryption and decryption.

> **i** Note
>
> The SAP Common Crypto Library is FIPS 140-2 compliant.

10.3 SSL Overview

SSL is an industry standard for sending wire- or socket-level encrypted data over secure network connections.

Before the SSL connection is established, the server and the client exchange a series of I/O round trips to negotiate and agree upon a secure encrypted session. This is called the SSL handshake.

SSL Handshake

When a client requests a connection, the SSL-enabled server presents its certificate to prove its identity before data is transmitted. Essentially, the handshake consists of the following steps:

- The client sends a connection request to the server. The request includes the SSL (or Transport Layer Security, TLS) options that the client supports.
- The server returns its certificate and a list of supported cipher suites, which includes SSL/TLS support options, algorithms used for key exchange, and digital signatures.
- A secure, encrypted session is established when both client and server have agreed upon a Cipher Suite.

For more specific information about the SSL Handshake and the SSL/TLS protocol, see the Internet Engineering Task Force Web site .

10.3.1 SSL in SAP ASE

The implementation of SSL provides several levels of security.

- The server authenticates itself—proves that it is the server you intended to contact—and an encrypted SSL session begins before any data is transmitted.
- Once the SSL session is established, the client requesting a connection can send his user name and password over the secure, encrypted connection.
- A comparison of the digital signature on the server certificate can determine whether the data received by the client was modified before reaching the intended recipient.

SAP ASE uses the proprietary SAP Common Crypto Library as the SSL provider and cryptography library. Beginning with SAP ASE version 16.0 SP03, the OpenSSL library is not supported for encryption and decryption.

SAP Common Crypto Library is FIPS 140-2 compliant. For information about FIPS certification, see the *Computer Security Resource Center* at https://csrc.nist.gov.

10.3.1.1 SSL Filter

The SAP ASE directory service, such as the `interfaces` file, Windows Registry, or LDAP service, defines the server address and port numbers, and determines the security protocols that are enforced for client connections. SAP ASE implements the SSL protocol as a filter that is appended to the master and query lines of the directory services.

The addresses and port numbers on which SAP ASE accepts connections are configurable, so you can enable multiple network and security protocols for a single server. Server connection attributes are specified with directory services, such as LDAP, or with the traditional Sybase `interfaces` file.

All connection attempts to a `master` or `query` entry in the `interfaces` file with an SSL filter must support the SSL protocol. A server can be configured to accept SSL connections and have other connections that accept clear text (unencrypted data), or use other security mechanisms.

For example, the `interfaces` file on UNIX that supports both SSL-based connections and clear-text connections looks like this:

```
SYBSRV1
master tcp ether myhostname myport1 ssl="CN=my_server_common_name.mydomain.com"
query  tcp ether myhostname myport1 ssl="CN=my_server_common_name.mydomain.com"
master tcp ether myhostname myport2
```

The SSL filter is different from other security mechanisms, such as Kerberos, which are defined with SECMECH (security mechanism) lines in the `interfaces` file (`sql.ini` on Windows).

10.3.1.2 Authentication Via the Certificate

The SSL protocol requires server authentication via a server certificate to enable an encrypted session. Likewise, when SAP ASE is functioning as a client during RPCs, there must be a repository of trusted CAs that a client connection can access to validate the server certificate.

The Server Certificate

Each SAP ASE server must have its own server certificate file that is loaded at start-up. The following is the default location for the certificates file, where `<servername>` is the name of the server as specified on the command line during start-up with the `-s` flag, or from the environment variable `$DSLISTEN`:

- UNIX – `$SYBASE/$SYBASE_ASE/certificates/servername.crt`
- Windows – `%SYBASE%\%SYBASE_ASE%\certificates\servername.crt`

The server certificate file consists of encoded data, including the server's certificate and the encrypted private key for the server certificate.

Alternatively, you can specify the location of the server certificate file when using `sp_ssladmin`.

> **i** Note
>
> To make a successful client connection, the common name in the certificate must match the server name in the `interfaces` file.

The CA Trusted Roots Certificate

The list of trusted CAs is loaded by SAP ASE at start-up from the trusted roots file. The trusted roots file is similar in format to a certificate file, except that it contains certificates for CAs known to SAP ASE. A trusted roots file is accessible by the local server in the following, where `<servername>` is the name of the server:

- UNIX – `$SYBASE/$SYBASE_ASE/certificates/servername.txt`
- Windows – `%SYBASE%\%SYBASE_ASE\certificates\servername.txt`

The trusted roots file is only used by SAP ASE when it is functioning as a client, such as when performing RPC calls or Component Integration Services (CIS) connections.

The system security officer adds and deletes CAs that are to be accepted by SAP ASE, using a standard ASCII-text editor.

> **⚠** Caution
>
> Use the system security officer role (sso_role) within SAP ASE to restrict access and execution on security-sensitive objects.

The `$SYBASE/$SYBASE_ASE/certificate/servername.crt` file is an SAP ASE server certificate and the `$SYBASE/$SYBASE_ASE/certificates/servername.txt` file is the CA trusted roots certificate that has signed the SAP ASE server certificate.

Both of these files are loaded by SAP ASE during start-up. The file `$SYBASE/config/trusted.txt` contains list of CA trusted roots certificate that are used by the client (there could be many CA trusted roots certificates). This file should contains the CA trusted root certificate that is present in `$SYBASE/$SYBASE_ASE/certificates/servername.txt`. Note that the file extension "crt" is for the server certificate, and "txt" is for the CA trusted roots certificates.

10.3.1.3 Connection Types

Various client-to-server and server-to-server connections are available.

Client Login to SAP ASE

Open Client applications establish a socket connection to SAP ASE similarly to the way that existing client connections are established. Before any user data is transmitted, an SSL handshake occurs on the socket when the network transport-level connect call completes on the client side and the accept call completes on the server side.

Server-to-Server Remote Procedure Calls

SAP ASE establishes a socket connection to another server for RPCs in the same way that existing RPC connections are established. Before any user data is transmitted, an SSL handshake occurs on the socket when the network transport-level connect call completes. If the server-to-server socket connection has already been established, the existing socket connection and security context is reused.

When functioning as a client during RPCs, SAP ASE requests the remote server's certificate during connection. SAP ASE then verifies that the CA that signed the remote server's certificate is trusted; that is to say, on its own list of trusted CAs in the trusted roots file. It also verifies that the common name in the server certificate matches the common name used when establishing the connection.

Companion Server and SSL

You can use a companion server to configure SAP ASE for failover. You must configure both the primary and secondary servers with the same SSL and RPC configuration. When connections fail over or fail back, security sessions are reestablished with the connections.

Open Client Connections

Component Integration Services, RepAgent, Distributed Transaction Management, and other modules in SAP ASE use Client-Library to establish connections to servers other than SAP ASE servers. The remote server is

authenticated by its certificate. The remote server authenticates the SAP ASE client connection for RPCs with user name and password.

10.3.2 Enabling SSL

SAP ASE determines which security service it will use for a port based on the interface file (`sql.ini` on Windows).

Procedure

1. Obtain an SAP ASE license for the SSL feature. You can use either the ASE_ASM or ASE_SECDIRS license.
2. Run the command:

   ```
   sp_configure "enable ssl", 1
   ```

   Check that the run value is 1 and there are no errors in the log file.
3. Request a signed server certificate for the SAP ASE server. See Request Certificates [page 300].
   a. Make sure certificate file has the certificate's "private key" encrypted, encoded, and appended to the file.
   b. The `commonName` of the subject certificate must match the exact case of the `servername` in the interfaces file. Optionally, you can specify the `commonName` in the certificate with the SSL filter. See Create Server Directory Entries [page 309].
4. Run the command:

   ```
   sp_ssladmin addcert, <certificate_path>, <password>
   ```
5. Install the server CA certificate in: `$SYBASE/$SYBASE_ASE/certificates/<servername>.txt`
6. Add the server entry to the interfaces file. See Create Server Directory Entries [page 309].
7. Copy CA certificates to the clients `trusted.txt` file location which defaults to `$SYBASE/config/trusted.txt`

Results

Unlike other security services, such as Kerberos, and NTLAN, SSL relies neither on the “Security” section of the Open Client/Open Server configuration file `libtcl.cfg`, nor on objects in `objectid.dat`.

The system administrator should consider memory use by SSL when planning for total physical memory. You need approximately 40K per connection (connections include user connections, remote servers, and network listeners) in SAP ASE for SSL connections. The memory is reserved and preallocated within a memory pool and is used internally by SAP ASE and SSL Plus libraries as requested.

10.3.3 Enabling SSL in the Backup Server

The security mechanism used by the Backup Server is based on server directory entries for the security service (if any) set in the `interfaces` file for UNIX (`$SYBASE/interfaces`) and Windows (`%SYBASE%\ini\sql.ini`) respectively.

Context

To enable the SSL security service in the Backup Server, use the following procedure:

Procedure

1. Obtain an SSL-enabled license for the SAP ASE server. You can use one of `ASE_ASM`, or `ASE_SECDIRS` licenses.

   > **i Note**
   >
   > The SAP ASE server must be SSL-enabled to successfully perform `dump` and `load` operations.

2. Connect to the Backup Server with the administrator account (defined by the configuration parameter `admin_user`, and set to `sa` by default), and run the command:

   ```
   set use_ssl true
   ```

3. Request a signed server certificate for the SAP ASE server. See Requesting and Authorizing SSL Certificates Using Keytool and ExportPrivateKey Utility [page 301].

   > **i Note**
   >
   > Ensure that the private key of the server certificate is encrypted, encoded, and appended to the certificate file.

4. Run the commands:

   ```
   set ssl_identity_file <certificate_path>
   set ssl_password <password>
   ```

5. Add the server entry to the interfaces file. See Create Server Directory Entries [page 309].

6. Append the CA certificates to the file `trusted.txt` (which defaults to the path `$SYBASE/config/trusted.txt`). To specify a custom path for `trusted.txt`, run the command:

   ```
   ssl_ca_file <custom path for trusted.txt>
   ```

Example: Set up SSL Using a Direct Connection to the Backup Server

An illustration of how to set up SSL using a direct connection to the Backup Server.

Example

We use two Backup Servers here – **BkpSvr001** and **BkpSvr002**, each with their own `commonName` and certificate (although they can use the same certificate as well).

To set up SSL using a direct connection to these Backup Servers, we boot them with a non-SSL port:

```
************************************************************
 BkpSvr001
        query tcp ether asestore1 8888 ssl="CN=BkpSvr001"
        master tcp ether asestore1 8888 ssl="CN=BkpSvr001"
        query tcp ether asestore1 8889
        master tcp ether asestore1 8889
 BkpSvr002
        query tcp ether asestore1 8892 ssl="CN=BkpSvr002"
        master tcp ether asestore1 8892 ssl="CN=BkpSvr002"
        query tcp ether asestore1 8891
        master tcp ether asestore1 8891
************************************************************
```

First, let's look at how to set up SSL using a direct connection to BkpSvr001.

If you boot the Backup Server without specifying a configuration file, run the command:

```
$BACKUPSERVER -M$SYBMULTBUF -I$DEVDIR/interfaces -SBkpSvr001 -e$DEVDIR/
backup.log
```

When the Backup Server boots for the first time, it creates the file `$SYBASE/BkpSvr001.cfg` from `defaultbs.cfg`:

```
****************************
$ cat $SYBASE/defaultbs.cfg
server_name=DEFAULT
admin_user=DEFAULT
admin_password=DEFAULT
interfaces_file=DEFAULT
errorlog_file= DEFAULT
sybmultbuf_binary=DEFAULT
library_path=DEFAULT
hosts_allow_file=DEFAULT
tape_config_file=DEFAULT
language=DEFAULT
character_set=DEFAULT
network_connections=DEFAULT
server_connections=DEFAULT
service_threads=DEFAULT
warn_level=DEFAULT
notify_lang_warn=DEFAULT
packet_size=DEFAULT
max_shared_memory=DEFAULT
allocation_threshold=DEFAULT
reserved_threshold=DEFAULT
pll_scan_threads=DEFAULT
compress_level=DEFAULT
use_ssl=DEFAULT
ssl_password=DEFAULT
ssl_identity_file=DEFAULT
```

```
ssl_version=DEFAULT
ssl_ca_file=DEFAULT
ssl_cipher=DEFAULT
****************************
```

Since SSL is not enabled by default, we cannot connect through the SSL port yet.

To configure and enable SSL on BkpSvr001, run the commands:

```
****************************************************************************
***
$ isql -Usa -P -SBkpSvr001
1> set admin_password Sapase123
2> go
Backup Server: 9.9.1.1: Configuration file '/FolderZ/User666/sapase/release/
BkpSvr001.cfg' has been
written and the previous version has been renamed to
'/FolderZ/User666/sapase/release/BkpSvr001.cfg.000000001'.
Backup Server: 9.18.1.1: Configuration option 'admin_password' changed. The
Backup Server does not
need to be rebooted for this option to take effect since the option is
dynamic.
1> set ssl_identity_file /usr/u/User666/tools/SSL/BkpSvr001.crt
2> go
Backup Server: 9.9.1.1: Configuration file '/FolderZ/User666/sapase/release/
BkpSvr001.cfg' has been
written and the previous version has been renamed to
'/FolderZ/User666/sapase/release/BkpSvr001.cfg.000000002'.
Backup Server: 9.17.1.1: Configuration option 'ssl_identity_file' changed.
The Backup Server needs
to be rebooted for this option to take effect since the option is static.
1> set ssl_ca_file /FolderZ/User666/sapase/release/config/trusted.txt
2> go
Backup Server: 9.9.1.1: Configuration file '/FolderZ/User666/sapase/release/
BkpSvr001.cfg' has been
written and the previous version has been renamed to
'/FolderZ/User666/sapase/release/BkpSvr001.cfg.000000003'.
Backup Server: 9.18.1.1: Configuration option 'ssl_ca_file' changed. The
Backup Server does not need
to be rebooted for this option to take effect since the option is dynamic.
1> set ssl_password sapsecure101
2> go
Backup Server: 9.9.1.1: Configuration file '/FolderZ/User666/sapase/release/
BkpSvr001.cfg' has been
written and the previous version has been renamed to
'/FolderZ/User666/sapase/release/BkpSvr001.cfg.000000004'.
Backup Server: 9.17.1.1: Configuration option 'ssl_password' changed. The
Backup Server needs
to be rebooted for this option to take effect since the option is static.
1> quit
****************************************************************************
***
```

To connect again using the new admin password that we created above, run the commands (it isn't mandatory, but recommended to change the password for `admin_user` while setting up SSL):

```
****************************************************************************
***
$ isql -Usa -PSapase123 -SBkpSvr001
1> show config
2> go
config_file=/FolderZ/User666/ase160sp03plx/release/BkpSvr001.cfg
server_name=BkpSvr001
admin_user=sa
admin_password=0x02c38047d4d41e84a0e0d8dfdd71b57a99ac5907ee9f5948c1d86175d7f76
c61
```

```
interfaces_file=/FolderZ/User666/ase160sp03plx/devices/interfaces
errorlog_file=/FolderZ/User666/ase160sp03plx/devices/backup.log
sybmultbuf_binary=/FolderZ/User666/ssbuilt_807871-1/aux/linuxamd64/sym64bit/
sybmultbuf
library_path=/FolderZ/User666/ase160sp03plx/release/ASE-16_0/lib
hosts_allow_file=/FolderZ/User666/ase160sp03plx/devices/hosts.allow
tape_config_file=/FolderZ/User666/ase160sp03plx/release/backup_tape.cfg
language=us_english
character_set=utf8
network_connections=25
server_connections=30
service_threads=48
warn_level=0
notify_lang_warn=0
packet_size=2048
max_shared_memory=48
shared_memory_per_stripe=1048576
allocation_threshold=40
reserved_threshold=85
pll_scan_threads=2
compress_level=0
use_ssl=FALSE
ssl_password=0x02ccb33950f95b0c47c0cf5842bf2d5667133d6e2e6462beccc90294ac88eba
e
ssl_identity_file=/usr/u/User666/tools/SSL/BkpSvr001.crt
ssl_version=TLSv1
ssl_ca_file=/FolderZ/User666/ase160sp03plx/release/config/trusted.txt
ssl_cipher=TLS_RSA_WITH_AES_256_GCM_SHA384,TLS_RSA_WITH_AES_128_GCM_SHA256,
TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_3DES_
EDE_CBC_SHA,TLS_RSA_WITH_RC4_128_SHA,TLS_RSA_WITH_RC4_128_MD5TLS_DHE_DSS_WITH_
3DES_EDE_CBC_SHA,TLS_DHE_DSS_WITH_RC4_128_SHA,TLS_DHE_RSA_WITH_3DES_EDE_CBC_SH
A,
TLS_RSA_WITH_DES_CBC_SHA,TLS_DHE_DSS_WITH_DES_CBC_SHA,TLS_DHE_RSA_WITH_DES_CBC
_SHA
*********************************************************************
*****
```

This is what the configuration file now looks like:

```
*********************************************************************
***
$ cat $SYBASE/BkpSvr001.cfg
server_name=BkpSvr001
admin_user=DEFAULT
admin_password=0x02c38047d4d41e84a0e0d8dfdd71b57a99ac5907ee9f5948c1d86175d7f76
c61
interfaces_file=DEFAULT
errorlog_file=DEFAULT
sybmultbuf_binary=DEFAULT
library_path=DEFAULT
hosts_allow_file=DEFAULT
tape_config_file=DEFAULT
language=DEFAULT
character_set=DEFAULT
network_connections=DEFAULT
server_connections=DEFAULT
service_threads=DEFAULT
warn_level=DEFAULT
notify_lang_warn=DEFAULT
packet_size=DEFAULT
max_shared_memory=DEFAULT
allocation_threshold=DEFAULT
reserved_threshold=DEFAULT
pll_scan_threads=DEFAULT
compress_level=DEFAULT
use_ssl=DEFAULT
```

```
ssl_password=0x02ccb33950f95b0c47c0cf5842bf2d5667133d6e2e6462beccc90294ac88eba
e
ssl_identity_file=/usr/u/User666/tools/SSL/BkpSvr001.crt
ssl_version=DEFAULT
ssl_ca_file=DEFAULT
ssl_cipher=DEFAULT
********************************************************************************
***
```

After you create the configuration file, you can boot the Backup Server without any parameters. If any parameter is passed, it overwrites the corresponding value in the Backup Server configuration file.

To enable SSL on BkpSvr001 from the command line, run the command:

```
******************************
$ $BACKUPSERVER --use_ssl=true &
$ isql -Usa -PSapase123 -SBkpSvr001
1> show config use_ssl
2> go
use_ssl=TRUE
******************************
```

At this point, the non-SSL listeners can be removed from the interfaces file, requiring users to connect only through SSL. This enables both isql and SAP ASE to act as SSL clients of an SSL-enabled Backup Server.

The CA root certificate must be present in the file `$SYBASE/config/trusted.txt`. The configuration option `ssl_ca_file` allows the Backup Server to change the location of the CA root certificate, and also (by default) of `$SYBASE/config/trusted.txt`, so that it can act as an SSL client when working as the client of a remote Backup Server, in a remote dump or load.

You have now successfully set up SSL on BkpSvr001.

To set up SSL on BkpSvr002, repeat the tasks you performed for BkpSvr001.

After you complete those tasks, run the following command:

```
2> dump database model to "/tmp/model.dmp" at BkpSvr002
3> go
```

As an option, you can specify the hostname (asestore1) and port (8892) with the command itself:

```
1> dump database model to "/tmp/model.dmp" at 'asestore1 8892
ssl="CN=BkpSvr002"'
2> go
```

You have successfully set up SSL using a direct connection to both Backup Servers, BkpSvr001 and BkpSvr002.

10.3.4 Certificate Request Overview

The system security officer installs server certificates and private keys for SAP ASE.

To generate a certificate, you can use a third party tool, such as the `openssl` utility, or use the `keytool` and `ExportPrivateKey` utility provided with SAP ASE.

To obtain a certificate, you must request a certificate from a certificate authority (CA). SAP ASE requires SSL certificates to use the PEM format. However, the certificate authority may deliver certificates in a format other than PEM. You must convert the certificate to the PEM format.

If you request a certificate from a third party and that certificate is in PKCS #12 format, use the `openssl` or any other third party tool to convert the certificate into a format that is understood by SAP ASE.

When FIPS is configured, MD5 hash algorithms cannot be used for certificate generation. The private key must be in PKCS #8 encrypted format.

The main steps to creating a certificate for use with SAP ASE are:

1. Generate the public and private key pair.
2. Securely store the private key.
3. Generate the certificate request.
4. Send the certificate request to the CA.
5. After the CA signs and returns the certificate, store it in a file and append the private key to the certificate.
6. Store the certificate in the SAP ASE installation directory.

Most third-party PKI vendors and some browsers have utilities to generate certificates and private keys. These utilities are typically graphical wizards that prompt you through a series of questions to define a distinguished name and a common name for the certificate.

Follow the instructions provided by the wizard to create certificate requests. Once you receive the signed PKCS #12-format certificate, use `openssl` tool or any other tool to generate a certificate file and a private key file. Concatenate the two files into a `servername.crt` file, where `<servername>` is the name of the server, and place it in the `certificates` directory under `$SYBASE/$SYBASE_ASE`.

10.3.5 Requesting and Authorizing SSL Certificates Using Keytool and ExportPrivateKey Utility

Use the `ExportPrivateKey` utility to encrypt a server private key in Public-Key Cryptography Standards (PKCS) 8 format.

To generate an SSL certificate, use keytool (a key and certificate management tool) with `ExportPrivateKey` utility. The process of creating an SSL certificate involves multiple steps. Some steps, such as generating a key pair, exporting certificate, and creating a root certificate require you to use the keytool, whereas encrypting a server private key in PKCS8 format requires the `ExportPrivateKey` utility.

The `ExportPrivateKey` utility is bundled with the SAP ASE installer and is installed at `$SYBASE/$SYBASE_OCS/bin`. The keytool is also provided with the SAP ASE installer. The utility can be used to create SSL certificates with Java Runtime Environment (JRE) version 7 or later.

> **i Note**
>
> While using the `ExportPrivateKey` utility with the keytool, it is recommended to provide the full directory path to the keytool.

After you create an SSL certificate, use the `sp_ssladmin addcert, <certificate_path>, <password>` system procedure to install the certificate. SAP ASE installs SSL certificates at `$SYBASE/$SYBASE_ASE/certificates`.

10.3.5.1 Creating a Self-signed Root Certificate

Generating a self-signed root certificate involves creating a key pair and exporting the certificate from the keystore to a text file.

Procedure

1. Create a key pair (public-private keys). The private key is then used to sign the certificate. Run the following command:

   ```
   keytool -genkeypair -alias ca -keyalg RSA -keysize 1024 -dname "cn=TestCA,
   o=SAP, l=Dublin, st=CA, c=US, ou=Support" -keypass sap123 -ext bc:c -validity
   3650 -keystore ca.jks -storepass sap123
   ```

   After the successful execution of this command, a key pair is created for CA and is stored in the keystore (`ca.jks`).
2. Export the certificate from the keystore (`ca.jks`) to a text file (`trusted.txt`). Run the following command:

   ```
   keytool -exportcert -alias ca -file trusted.txt -keystore ca.jks -storepass
   sap123 -rfc
   ```

   After the successful execution of this command, the certificate is exported to a text file.
3. Create the trusted roots file, `server.txt`:
   - UNIX:

     ```
     cp trusted.txt $SYBASE/$SYBASE_ASE/certificates/server.txt
     ```
   - Windows:

     ```
     copy trusted.txt %SYBASE%\%SYBASE_ASE%\certificates\server.txt
     ```

10.3.5.1.1 Generate Key Pair Using genkeypair Option

Use the genkeypair option of keytool to generate a key pair and store it in the keystore.

A sample command to generate a key pair is as follows:

```
keytool -genkeypair -alias ca -keyalg RSA -keysize 1024 -dname "cn=TestCA,
o=SAP, l=Dublin, st=CA, c=US, ou=Support" -keypass sap123 -ext bc:c -validity
3650 -keystore ca.jks -storepass sap123
```

Where

- `-genkeypair` – The keytool command to generate a public key and an associated private key.
- `-alias` – The alias (`ca`) to access the keystore (`ca.jks`). The keystore stores the key pair. Each entry in the keystore is identified by a unique name or an alias.

- `-keyalg` – The name of the algorithm (`RSA`) used to generate the key pair.
- `-keysize` – The size (`1024` bits) of the generated key pair.
- `-dname` – Specifies the X.500 distinguished name to be associated with the alias. It is used as the issuer and subject fields in the self-signed certificate. In this example, the values are as follows:
  - (common name) `cn=TestCA`
  - (organization) `o=SAP`
  - (locality) `l=Dublin`
  - (state) `st=CA`
  - (country) `c=US`
  - (organization unit) `ou=Support`
- `-keypass` – The password (`sap123`) to recover the key pair. If you do not specify the option on the command line, keytool will first attempt to use the keystore password to recover the private key, and if it fails, will then prompt you for the private key password.
- `-ext` – Denotes an X.509 certificate extension (BasicContraints:critical).
- `-validity` – Certificate validity (`3650` days)
- `-keystore` – The keystore (`ca.jks`) that stores the key pair.
- `-storepass` – The password (`sap123`) to access the keystore.

After the successful execution of this command, a key pair is created for the CA and stored in the keystore (`ca.jks`).

10.3.5.1.2 Export Certificate Using exportcert Option

Use the exportcert option of keytool to export a certificate from the keystore.

A sample command to export a certificate from the keystore (`ca.jks`) to a text file (`trusted.txt`) is as follows:

```
keytool -exportcert -alias ca -file trusted.txt -keystore ca.jks -storepass
sap123 -rfc
```

Where

- `-exportcert` – The keytool command to export a certificate.
- `-alias` – The alias (`ca`) to access the keystore (`ca.jks`).
- `-file` – The text file (`trusted.txt`) that contains the exported certificate.
- `-keystore` – The keystore (`ca.jks`) that stores the key pair.
- `-storepass` – The password (`sap123`) to access the keystore (`ca.jks`).
- `-rfc` – Indicates that the certificate is exported in PEM format.

10.3.5.2 Creating a Private Key and a Certificate Request for Server

Generate a key pair and a certificate request for a server.

Procedure

1. Create a key pair for the server by running the following command:

   ```
   keytool -genkeypair -alias server -keyalg RSA -keysize 1024 -dname
   "cn=<server>, o=SAP, l=Dublin, st=CA, c=US" -keypass sybase -ext bc:c -
   validity 3650 -keystore server.jks -storepass sybase
   ```

   After the successful execution of this command, a key pair is created for the server and is stored in the server keystore as `server.jks`.
2. Generate a certificate request for a server. Run the following command:

   For Open Server 15.7 and later version:

   ```
   keytool -certreq -alias server -file <server>.req -keypass sybase -keystore
   server.jks -storepass sybase
   ```

   For Open Server 15.5 and earlier versions:

   ```
   keytool -certreq -alias server -sigalg SHA1withRSA -file <server>.req -
   keypass sybase -keystore server.jks -storepass sybase
   ```

   A successful execution of this command generates a certificate request file (`file <server>.req`).

10.3.5.2.1 Generate Key Pair For Server Using genkeypair Option

Use the genkeypair option of keytool to generate a key pair for the server and store the key pair in the server keystore.

A sample command to generate a key pair is as follows:

```
keytool -genkeypair -alias server -keyalg RSA -keysize 1024 -dname "cn=<server>,
o=SAP, l=Dublin, st=CA, c=US" -keypass sybase -ext bc:c -validity 3650 -keystore
server.jks -storepass sybase
```

Where

- `-genkeypair` – The keytool command to generate a public key and an associated private key.
- `-alias` – The alias (`server`) to access the keystore (`server.jks`). The keystore stores the key pair. Each entry in the keystore is identified by a unique name or an alias.
- `-keyalg` – The name of the algorithm (`RSA`) used to generate the key pair.

- `-keysize` – The size (1024 bits) of the generated key pair.
- `-dname` – Specifies the server (`server`) for which a key pair is being generated. Other values are as follows:
  - (common name) `cn=server`
  - (organization) `o=SAP`
  - (locality) `l=Dublin`
  - (state) `st=CA`
  - (country) `c=US`
  - (organization unit) `ou=Support`
- `-keypass` – The password (`sybase`) to recover the key pair. If you do not specify the option on the command line, keytool will first attempt to use the keystore password to recover the private key, and if it fails, will then prompt you for the private key password.
- `-ext` – Denotes an X.509 certificate extension (BasicContraints:critical).
- `-validity` – Certificate validity (`3650` days)
- `-keystore` – The keystore (`server.jks`) that stores the server key pair.
- `-storepass` – The password (`sybase`) to access the server keystore.

A successful execution of this command creates a key pair for the server, which is stored in the server keystore (`server.jks`).

10.3.5.2.2 Generate a Certificate Request For Server Using certreq Option

Use the certreq option of keytool to generate a certificate request for the server.

A sample command to generate a certificate request for the server is as follows:

```
keytool -certreq -alias server -file <server>.req -keypass sybase -keystore
server.jks -storepass sybase
```

Where

- `-certreq` – The keytool command to generate a certificate request for a server.
- `-alias` – The alias (`server`) to access the server keystore (`server.jks`).
- `-file <server>.req` – Generated certificate request file for the server.
- `-keypass` – The password (`sybase`) to access the key pair.
- `-keystore` – The server keystore (`server.jks`).
- `-storepass` – The password (`sybase`) to access the server keystore.

10.3.5.3 Converting a Certificate Request to a CA-signed Server Certificate

Generate a server certificate signed by CA.

Procedure

Generate a certificate as a response to a certificate request file by running the following command:

For Open Server 15.7 and later version:

```
keytool -gencert -rfc -infile <server>.req -outfile <server>.crt -alias ca -ext
bc:c -keystore ca.jks - storepass sap123
```

For Open Server 15.5 and earlier versions:

```
keytool -gencert -rfc -infile <server>.req -outfile <server>.crt -alias ca -
sigalg SHA1withRSA -ext bc:c -keystore ca.jks -storepass sap123
```

10.3.5.3.1 Convert a Certificate Request to a CA-signed Server Certificate

Use the gencert option of keytool to accept a certificate request file as an input and generate a server certificate signed by a CA.

A sample command to generate a CA-signed certificate is as follows:

```
keytool -gencert -rfc -infile <server>.req -outfile <server>.crt -alias ca -ext
bc:c -keystore ca.jks -storepass sap123
```

Where

- `-gencert` – Generate a certificate as a response to a certificate request file.
- `-infile` – The input file is the certificate request file (`<server>.req`).
- `-outfile` – The output file is the server certificate (`<server>.crt`).
- `-alias` – The alias (`ca`) to access the keystore.
- `-ext` – Denotes an X.509 certificate extension (`bc:c`).
- `-keystore` – The CA keystore (`ca.jks`).
- `-storepass` – The password (`sap123`) to access the CA keystore.

10.3.5.4 Encrypting the Server Private Key in PKCS8 Format

Use the `ExportPrivateKey.jar` utility to extract the server private key from the server keystore and then encrypt the key in Public-Key Cryptography Standards (PKCS) 8 format.

Procedure

Set `<CLASSPATH>` environment variable to `$SYBASE/$SYBASE_OCS/bin/ExportPrivateKey.jar`, and run the following command:

```
java com.sap.security.ExportPrivateKey -keystore server.jks -alias server -file
<server>.prk -storepass sybase
```

10.3.5.4.1 Encrypt the Server's Private Key in PKCS8 Format

Use the `ExportPrivateKey` utility to encrypt a server's private key in PKCS8 format.

```
java com.sap.security.ExportPrivateKey -keystore server.jks -alias server -file
<server>.prk -storepass sybase
```

Where

- `java com.sap.security.ExportPrivateKey` – The command to encrypt the server private key in PKCS8 format using the `ExportPrivateKey` utility.
- `-keystore` – The server keystore (`server.jks`).
- `-alias` – The alias (`server`) to access the keystore.
- `-file` – The generated file that stores the encrypted private key.
- `-storepass` – The password (`sybase`) to access the server keystore.

10.3.5.5 Appending Private Key to the Server Certificate

Append the generated private key to the server certificate.

Procedure

Run the following command to append the generated private key (`<server>.prk`) to the server certificate (`<server>.crt`).

```
cat <server>.prk >> <server>.crt
```

10.3.5.6 ExportPrivateKey

Use the `ExportPrivateKey` utility to encrypt a server private key in Public-Key Cryptography Standards (PKCS) 8 format. To generate an SSL certificate, you need to use keytool (a key and certificate management tool) with the `ExportPrivateKey` utility. The process of creating an SSL certificate involves multiple steps. Some steps, such as generating a key pair, exporting certificate, and creating a root certificate require you to use the keytool, whereas encrypting a server private key in PKCS8 format requires the `ExportPrivateKey` utility.

Syntax

```
ExportPrivateKey
[-keystore <server_keystore>]
[-alias <alias>]
[-file <server_file.prk>]
[-storepass <password>]
[-keypass <password>]
```

Parameters

`-keystore <server_keystore>`

Specifies the server keystore.

`-alias <alias>`

Specifies the alias to access the keystore.

`-file <server_file.prk>`

File that stores the encrypted private key.

`-storepass <password>`

Password to access the keystore.

`-keypass <password>`

Specifies a password that is different from the one specified for `keystore`. The password used for the keystore is used to access the private key if you do not include `keypass`,

Example

In this example, the `ExportPrivateKey` utility extracts the server private key from the server keystore (`<server.jks>`) using the alias (`<server>`), and then encrypt the key and stores it in the `<server.prk>` file. It uses `<sap123>` as the password to access the server keystore.

```
java com.sap.security.ExportPrivateKey -keystore <server.jks> -alias <server> -
file <server.prk> -storepass <sap123>
```

10.3.6 Create Server Directory Entries

SAP ASE accepts client logins and server-to-server RPCs. The address and port numbers where SAP ASE accepts connections are configurable so you can specify multiple networks, different protocols, and alternate ports.

In the `interfaces` file, SSL is specified as a filter on the master and query lines. The following example shows an entry for an server using SSL in a UNIX environment:

```
[SYBSRV1]
master tcp ether myhostname  myport ssl="CN=my_server_common_name.mydomain.com"
query  tcp ether myhostname   myport ssl="CN=my_server_common_name.mydomain.com"
```

In this example, the SSL security service is specified on port number `myport`.

An entry for the server with SSL and Kerberos security mechanisms on Windows might look like:

```
[SYBSRV2]
    query=nlwnsck, 18.52.86.120,2748,ssl
    master=nlwnsck 18.52.86.120,2748,ssl
    master=nlwnsck 18.52.86.120,2749
    secmech=1.3.6.1.4.897.4.6.6
```

In this example, the SSL security service is specified on port number `2748(0x0abc)`.

The SECMECH line in the example contains an object identifier (OID) that refers to the security mechanism Kerberos, respectively. The OID values are defined in:

- UNIX – `$SYBASE/$SYBASE_OCS/config/objectid.dat`
- Windows – `%SYBASE%\%SYBASE_OCS\ini\objectid.dat`

> **i Note**
>
> The use of SSL concurrently with a SECMECH security mechanism is intended to facilitate migration from SECMECHs to SSL security.

10.3.7 Administer Certificates

To administer SSL and certificates in SAP ASE, use `sp_ssladmin`. `sso_role` is required to execute the stored procedure.

`sp_ssladmin` is used to:

- Add local server certificates. You can add certificates and specify the password used to encrypt private keys, or require input of the password at the command line during start-up.
- Delete local server certificates.
- List server certificates.

For example:

```
sp_ssladmin addcert, "/sybase/ASE-16_0/certificates/<Server1.crt>",
      "<mypassword>"
```

This adds an entry for the local server, `<Server1.crt>`, in the certificates file in the absolute path to `$SYBASE/ASE-16_0/certificates`. The private key is encrypted with the password "`<mypassword>`". The password should be the one specified when you created the private key.

Before accepting the certificate, `sp_ssladmin` verifies that:

- The private key can be decrypted using the provided password (except when NULL is specified).
- The private key and public key in the certificate match.
- The certificate chain, from root CA to the server certificate, is valid.
- The common name in the certificate matches the common name in the `interfaces` file.

If the common names do not match, `sp_ssladmin` issues a warning. If the other criteria fails, the certificate is not added to the certificates file.

> ⚠ Caution
>
> Passwords are limited to 64 characters.

The use of NULL as the password is intended to protect passwords during the initial configuration of SSL, before the SSL-encrypted session begins. Since you have not yet configured SSL, the password travels unencrypted over the connection. You can avoid this by specifying the password as NULL during the first login.

When NULL is the password, you must start `dataserver` with a `-y` flag, which prompts the administrator for the private-key password at the command line.

After restarting the server with an SSL connection established, use `sp_ssladmin` again, this time using the actual password. The password is then encrypted and stored by SAP ASE. Any subsequent starts of the server from the command line use the encrypted password; you do not have to specify the password on the command line during start-up.

An alternative to using a NULL password during the first login is to avoid a remote connection to SAP ASE via `isql`. You can specify "localhost" as the `<hostname>` in the `interfaces` file (`sql.ini` on Windows) to prevent clients from connecting remotely. Only a local connection can be established, and the password is never transmitted over a network connection.

SAP ASE has sufficient memory in its network memory pool to allow `sp_ssladmin addcert` to set the certificate and private key password with its default memory allocations. However, if another network memory

consumer has already allocated the default network memory, `sp_ssladmin` may fail and display this error to the client:

```
Msg 12823, Level 16, State 1:
Server 'servername', Procedure 'sp_ssladmin', Line 72:
Command 'addcert' failed to add certificate path
/work/REL125/ASE-16_0/certificates/servername.crt, system error: ErrMemory.
(return status = 1)
```

Or the following message may appear in the error log:

```
 ... ssl_alloc: Cannot allocate using ubfalloc(rnetmempool, 131072)
```

As a workaround, you can increase the `additional network memory` configuration parameter. About 500K bytes of memory is needed for `sp_ssladmin addcert` to succeed, so increasing additional network memory by this amount may allow it to succeed. This memory is reused by the network memory pool when needed, or you can return `additional network memory` to its previous value after `sp_ssladmin` has successfully completed.

See *Reference Manual: Procedures* for a complete `sp_ssladmin` syntax.

10.3.7.1 Updating an Expired Certificate

Use `sp_ssladmin` to replace an expired certificate on your local server.

Procedure

1. Use the `dropcert` parameter of the `sp_ssladmin` system stored procedure to drop the expired certificate. For example:

   ```
   sp_ssladmin dropcert , "/sybase/ASE-16_0/certificates/Server1.crt"
   ```

2. Use the `addcert` parameter to add a certificate, for example:

   ```
   sp_ssladmin addcert, "/sybase/ASE-16_0/certificates/Server1_new.crt",
   "mypassword"
   ```

3. Activate the certificate:

   ```
   sp_ssladmin refreshcert
   ```

10.3.8 Performance

There is additional overhead required to establish a secure session, because data increases in size when it is encrypted, and it requires additional computation to encrypt or decrypt information.

The additional memory requirements for SSL increases the overhead by 50-60 percent for network throughput or for establishing a connection. You must have approximately 40K more memory for each user connection.

10.3.9 Cipher Suites

During the SSL handshake, the client and server negotiate a common security protocol via a Cipher Suite. Cipher Suites are preferential lists of key-exchange algorithms, hashing methods, and encryption methods used by SSL-enabled applications.

For a complete description of Cipher Suites, visit the Internet Engineering Task Force (IETF) organizationhttp://tools.ietf.org/html/rfc5246 .

By default, the strongest Cipher Suite supported by both the client and the server is the Cipher Suite that is used for the SSL-based session.

> **i** Note
>
> The Cipher Suites listed conform to the Transport Layer Specification (TLS). TLS is an enhanced version of SSL 3.0, and is an alias for the SSL version 3.0 Cipher Suites.

10.3.9.1 Display or Set SSL Cipher Suites

You can display, set, or restrict the set of cipher suites that SAP ASE uses.

Display Cipher Suite Preferences

You can display cipher suite preferences that have been set using `sp_ssladmin lsciphers`. If no preferences have been set, `sp_ssladmin lsciphers` returns 0 rows to indicate no preferences are set and SAP ASE uses its default (internal) preferences.

The Transact-SQL global variable `<@@ssl_ciphersuite>` can be used to display the cipher suite that was chosen by the SSL handshake and verify that an SSL or a non-SSL connection was established.

For example, an `isql` connection using SSL protocol displays the cipher suite chosen for it.

```
1> select @@ssl_ciphersuite
2> go
------------------------------
TLS_RSA_WITH_AES_128_CBC_SHA
(1 row affected)
```

SAP ASE sets `<@@ssl_ciphersuite>` when the SSL handshake completes. The value is either NULL, indicating a non-SSL connection, or a string containing the name of the cipher suite chosen by the SSL handshake.

Set Cipher Suite Preferences

The set of cipher suites that SAP ASE uses can be restricted, giving control to the system security officer over the types of encryption algorithms that can be used by client connections to the server, or outbound connections from the SAP ASE server. SAP ASE uses an internally defined set of preferences for cipher suites.

To set a specific cipher suite preference, enter:

```
sp_ssladmin setciphers, { "TLS1.2" | "TLS1.0" | "FIPS" | "Strong" | "Weak" |
"All" | quoted_list_of_ciphersuites }
```

`sp_ssladmin setciphers` sets cipher suite preferences for a given ordered list. This restricts the available SSL cipher suites to the specified set of `"TLS1.2"`, `"TLS1.0"`, `"FIPS"`, `"Strong"`, `"Weak"`, `"All"`, or a quoted list of cipher suites. This takes effect on the next listener started, and requires that you restart the server to ensure that all listeners use the new settings.

Non-zero numbers indicate the preference order that SAP ASE uses during the SSL handshake. The client side of the SSL handshake chooses one of these cipher suites that matches its list of accepted cipher suites. An output preference of 0 (zero) indicates a cipher suite is not used by SAP ASE.

The following example specifies the set of cipher suites that use FIPS algorithms.

```
1> sp_ssladmin setcipher, 'FIPS'
```

```
The following cipher suites and order of preference are set for SSL connections:
Cipher Suite Name                                                Preference
---------------------------------------------------------------- -----------
TLS_RSA_WITH_AES_256_GCM_SHA384                                    1
TLS_RSA_WITH_AES_128_GCM_SHA256                                    2
TLS_RSA_WITH_AES_256_CBC_SHA                                       3
TLS_RSA_WITH_AES_128_CBC_SHA                                       4
```

This example sets a quoted list of cipher suite preferences:

```
1> sp_ssladmin setciphers, 'TLS_RSA_WITH_AES_128_CBC_SHA,
TLS_RSA_WITH_AES_256_CBC_SHA'
2> go
```

```
The following cipher suites and order of preference are set for SSL connections:
Cipher Suite Name                                                Preference
---------------------------------------------------------------- -----------
TLS_RSA_WITH_AES_128_CBC_SHA                                       1
TLS_RSA_WITH_AES_256_CBC_SHA                                       2
```

The system security officer should consider the security policies employed at his or her site and the available SSL cipher suites to decide whether to restrict cipher suites and which cipher suites are appropriate for the security policies.

For more information, see *Reference Manual: Procedures > sp_ssladmin*.

Remove Cipher Suite Preferences

If you have set SSL cipher suite preferences and want to remove all preferences from the server and use default preferences, delete the preferences from their storage location in system catalogs using the following commands:

```
1> sp_configure 'allow updates to system tables', 1
2> go
1> delete from master..sysattributes where class=24
2> go
1> sp_configure 'allow updates to system tables', 0
2> go
```

These commands can be executed only by the system security officer or system administrator.

Attempts to use any dropped cipher suite results in an SSL handshake failure and a failure to connect to SAP ASE.

10.3.9.2 Predefined Cipher Suites in SAP ASE

SSL cipher suites use an internally defined set of preferences for cipher suites.

The logical groups for cipher suite are as follows:

Logical Group	Cipher Suite
FIPS	• TLS_RSA_WITH_AES_256_GCM_SHA384 • TLS_RSA_WITH_AES_128_GCM_SHA256 • TLS_RSA_WITH_AES_256_CBC_SHA • TLS_RSA_WITH_AES_128_CBC_SHA
TLS 1.2	• TLS_RSA_WITH_AES_256_GCM_SHA384 • TLS_RSA_WITH_AES_128_GCM_SHA256
Strong	• TLS_RSA_WITH_AES_256_GCM_SHA384 • TLS_RSA_WITH_AES_128_GCM_SHA256 • TLS_RSA_WITH_AES_256_CBC_SHA • TLS_RSA_WITH_AES_128_CBC_SHA
TLS 1.0	• TLS_RSA_WITH_AES_256_CBC_SHA • TLS_RSA_WITH_AES_128_CBC_SHA • TLS_RSA_WITH_3DES_EDE_CBC_SHA • TLS_RSA_WITH_RC4_128_SHA • TLS_RSA_WITH_RC4_128_MD5
Weak	• TLS_RSA_WITH_3DES_EDE_CBC_SHA • TLS_RSA_WITH_RC4_128_SHA • TLS_RSA_WITH_RC4_128_MD5

Logical Group	Cipher Suite
All	• TLS_RSA_WITH_AES_256_GCM_SHA384 • TLS_RSA_WITH_AES_128_GCM_SHA256 • TLS_RSA_WITH_AES_256_CBC_SHA • TLS_RSA_WITH_AES_128_CBC_SHA • TLS_RSA_WITH_3DES_EDE_CBC_SHA • TLS_RSA_WITH_RC4_128_SHA • TLS_RSA_WITH_RC4_128_MD5

> **i** Note
>
> Diffie-Hellman Ephemeral Digital Signature Standard (DHE_DSS) is no longer supported for DSA certificates and cipher suites.

SAP ASE version 16.0 SP03 and later no longer supports these cipher suites:

- TLS_RSA_WITH_DES_CBC_SHA
- TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_RC4_128_SHA
- TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_DHE_DSS_WITH_DES_CBC_SHA
- TLS_DHE_RSA_WITH_DES_CBC_SHA
- TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
- TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
- TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA
- TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
- TLS_RSA_EXPORT_WITH_RC4_40_MD5
- TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
- TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
- TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
- TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA
- TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
- TLS_RSA_WITH_NULL_SHA
- TLS_RSA_WITH_NULL_MD5

10.3.10 Display or Set the TLS Protocol

You can display or set the TLS protocol version for client connections.

Display the TLS Protocol

To display the TLS protocol version, use `sp_ssladmin lsprotocol` or `@@ssl_protocol`.

In this example, a value of 0 is returned by `sp_ssladmin lsprotocol` when the protocol is set to NULL:

```
1> sp_ssladmin lsprotocol
2> go
TLS Protocol Version
 --------------------
  0
(1 row affected)
(return status = 0)
```

In this example, the Transact-SQL global variable `@@ssl_protocol` is used to identify that version TSL1.2 is negotiated by client and server:

```
1> select @@ssl_protocol
2> go
----------
TLSv1.2
(1 row affected)
```

Set the TLS Protocol

Set the TLS protocol version to `TLS1.2` or `NULL` using `sp_ssladmin setprotocol`. When you set the TLS protocol to `TLS 1.2`, the client must use the TLS 1.2 protocol version to successfully connect. Setting `sp_ssladmin setprotocol` to `NULL` allows backward compatibility by accepting TLS 1.0 and TLS 1.2 protocol versions from clients (this is the default value).

For example:

```
1> sp_ssladmin setprotocol,'TLS1.2'
2> go
 TLS Protocol Version
 --------------------
 TLSv1.2
(1 row affected)
(return status = 0)
1> sp_ssladmin lsprotocol
2> go
 TLS Protocol Version
 --------------------
 TLSv1.2
(1 row affected)
(return status = 0)
```

> **i** Note
>
> Cipher suites are independent of the TLS protocol version.

For more information, see *Reference Manual: Procedures > sp_ssladmin*.

10.3.11 Use SSL to Specify a Common Name

The server name specified in the directory service entry can be different from the common name the SSL server certificate uses to perform an SSL handshake.

This allows you to use a fully-qualified domain name for the SSL certificate common name (for example, `server1.bigcompany.com`).

To add a common name to the interfaces file, use:

```
ase1
   master tcp ether <host_name> <port_number> ssl="CN='<common_name>'"
   query tcp ether <host_name> <port_number> ssl="CN='<common_name>'"
```

When clients use SSL to connect to an SAP ASE server that also uses SSL, the SSL filter is placed after the port number in the `interfaces` file. The directory service includes the common name, which you add either by using `dsedit` or a text editor.

10.3.11.1 Specify a Common Name with sp_listener

`sp_listener` includes the `CN=<common_name>` parameter, which allows you to specify a common name for the SSL certificate.

The syntax is:

```
sp_listener 'command','[protocol:]<machine_name>:<port_number>:
"CN=<common_name>"', '<engine_number>'
```

Where `CN=<common_name>` is used only if you specify `ssltcp` as the protocol. The `<common_name>` you specify here is validated against the `<common_name>` in the SSL certificate. If you do not include `CN=<common_name>`, SAP ASE uses `<server_name>` to validate against the common name in the SSL certificate. If you include a fully-qualified domain name in the certificate, it must match the `CN=<common_name>`.

The attribute name "CN" is case insensitive (it can be "CN", "cn" or "Cn"), but the attribute value for the common name is case sensitive.

For example, to specify the common name `ase1.big server 1.com`:

```
sp_listener 'start','ssltcp:blade1:17251:"CN=ase1.big server 1.com"','0'
```

See the *Reference Manual: Procedures* for more information about `sp_listener`.

10.3.12 Using isql to Connect to an SSL-enabled SAP ASE Server

You can connect to an SSL-enabled SAP ASE server.

Procedure

Issue a command similar to:

```
isql -Usa -Pbigsecret96 -S'myhost 5000 ssl="CN=awsase1"'
```

where:

- <user name> – sa
- <password> – bigsecret96
- <host name> – myhost
- <port number> – 5000
- The ASE server common name (CN) as used in the SSL server certificate – awsase

10.3.13 Viewing a Certificate Expiration Date

Use the `monSSLCertInfo` monitoring table to view the expiration date of the SSL certificate loaded into SAP ASE.

Context

You must add the SSL certificate in SAP ASE server and activate the certificate first before you can view the certificate details.

Procedure

1. Use the `sp_ssladmin addcert` stored procedure to add the SSL certificate for the SAP ASE server:

   ```
   sp_ssladmin addcert, <certificate_path>,<password>
   ```

   - <certificate_path>
     Specify the absolute path to the certificates file on the local server.
   - <password>
     The password used to encrypt the private key when adding the new server certificate to the certificates file.

This example adds the SSL certificate path `/sybase/ASE-16_0/certificates/ase1.crt` for the SAP ASE server:

```
sp_ssladmin addcert,'/sybase/ASE-16_0/certificates/ase1.crt',<password>
go
```

2. Activate the SSL certificate on the SAP ASE server:

```
sp_ssladmin refreshcert
go
```

3. View certificate details in the `monSSLCertInfo` monitoring table:

```
select * from monSSLCertInfo
go
```

The result is:

```
 CertificatePath                   ValidFrom           ValidTo
 ----------------------------------- -------------------
-------------------
 /sybase/ASE-16_0/certificates/ase1.crt Aug 19 2016 11:58AM May 16 2019
11:58AM
(1 row affected)
```

You can also view the certificate valid dates in the `errorlog` file once the certificate is activated, or after you restart the server. For example:

```
00:0006:00000:00015:2019/02/06 21:47:45.97 kernel
SSL Certificate '/sybase/ASE-16_0/certificates/ase1.crt'
with validity from 'Aug 19 2016 11:58AM' up to 'May 16 2019 11:58AM'
is loaded successfully.
```

10.4 Encryption

SAP ASE comes with several encryption options to allow you to comply with security and privacy requirements. The differing methods allow you to choose whichever feature is easiest for you to deploy.

SAP ASE supports these forms of encryption:

Type	Description
Database encryption	A fully encrypted database encrypts all of its data, indexes, and transaction logs. This is done transparently, so you can perform operations as usual.
Column encryption	You can encrypt table columns by adding the `encrypt column` qualifier to the `create table` statement.
Session-level encryption using `isql`	Executing the `--command_encrypt` command line option encrypts the commands for the full active `isql`

Type	Description
	session. Executing the `--conceal` command line option hides and encrypts user input during the `isql` session.
Command-level encryption using `isql`	Adding `encrypt` to the command executor `go`encrypts commands on a per-command basis during an `isql` session.
Bulk copy encryption using the `bcp` utility	In versions of SAP ASE 16.0 SP03 and later, `bcp` automatically encrypts the `initstring`, `colpasswd`, and `keypasswd` options.
Encrypting commands between servers using CIS	Executing the `set remote command encryption {on|off}` command activates encryption for those commands that process sensitive data during the active command session.

For more information, see *Database Encryption* > *Overview of Encryption* and *Utility Guide* > *bcp and Encryption*.

10.4.1 On-Demand Encryption

On-demand encryption allows you to issue encrypted commands and system procedures as needed, for a full `isql` session, and on an ad hoc basis when you want to run individual commands that contain sensitive data.

When you use on-demand encryption, the client encrypts the command before sending it to the server. The server decrypts the command before processing it. On-demand encryption requires both the client and server to run SAP ASE versions 16.0 SP03 or later, or the system returns an error. No permissions are required to use on-demand encryption.

10.4.1.1 @@prev_batch_encrypted

Use the `<@@prev_batch_encrypted>` global variable to find out whether your previously executed commands were sent encrypted.

This global variable checks the encryption status of the most recently executed batch of commands in the current user session. A return of 0 indicates that the server received the command as plain text; a return of 1 indicates that the commands were received encrypted.

This example encrypts a command to create a login named "test_login," with successful encryption indicated by the return value "1":

```
create login test_login with password '<password>'
go encrypt
select @@prev_batch_encrypted
go
```

```
    1
```

This example shows the same command and is not encrypted, which is indicated by the return value "0":

```
create login test_login with password '<password>'
go
select @@prev_batch_encrypted
go
    0
```

10.4.1.2 Session-Level Encryption

Encrypt all of your commands in a session using the `--command_encryption` or `--conceal` command line options.

Specifying the `--command_encryption` option encrypts every command sent during the `isql` session.

In this example, the guest login encrypts the entire `isql` session:

```
isql -Uguest -P<password> -Smyase --command_encryption
```

Specifying the `--conceal` option hides and encrypts your input during the `isql` session.

For example:

```
isql -Uguest -P<password> -Smyase --conceal
```

Use `--command_encryption` when the `isql` session contains sensitive data.

Use `--conceal` when inputting visually sensitive data (such as passwords).

10.4.1.3 Command-Level Encryption

You can encrypt individual commands by adding `encrypt` to the `go` command terminator for `isql`.

Add `encrypt` after the `go` command terminator to encrypt the command you just specified. Only that command is encrypted; no subsequent command sent by `go` is encrypted unless you add `encrypt`.

For example, this encrypted command modifies the password for a login from "master" to "password123":

```
alter login test_login with password 'master' modify password 'password123'
go encrypt
```

10.4.1.3.1 List of Commands and System Procedures that Transmit Sensitive Data

Use on-demand encryption to encrypt commands transmitting sensitive data.

A number of commands transmit data that your site may deem sensitive, and should be encrypted.

- `alter encryption key`
- `alter login`
- `alter role`
- `connect`
- `create cluster`
- `create encryption key`
- `create login`
- `create role`
- `deploy plugin`
- `drop encryption key`
- `dump database`
- `dump transaction`
- `load database`
- `load transaction`
- `set cluster`
- `set encryption passwd`
- `set role`
- `show agent`
- `upgrade server`
- `sp_addexternlogin`
- `sp_addlogin`
- `sp_companion`
- `sp_encryption`
- `sp_extrapwdchecks`
- `sp_ldapadmin`
- `sp_password`
- `sp_ssladmin`

10.4.2 Encrypting Commands Between Local and Remote Servers Using CIS

CIS supports encryption for some commands and system procedures between servers by enabling command encryption for the session.

To encrypt commands sent between the servers, use the following `set` command for the active session:

```
set remote command encryption {on|off}
```

You can use this command as long as both the local and remote servers use SAP ASE version 16.0 SP03 or higher; the client need not use this version.

This encrypts the following:

- `connect to <server_name>`
- `execute <server_name>...<proc_name>`
- `sp_autoconnect`
- `sp_passthru`
- `sp_remotesql`

If one of the servers does not support encryption, the command is sent unencrypted.

For more information, see *Reference Manual: Commands* > *set*, and the *Component Integration Services Users Guide*.

10.5 Kerberos Confidentiality

To require all messages into and out of SAP ASE to be encrypted, set the `msg confidentiality reqd` configuration parameter to 1.

If this parameter is 0 (the default), message confidentiality is not required but may be established by the client.

For example, to require that all messages be encrypted, execute:

```
sp_configure "msg confidentiality reqd", 1
```

10.6 Dump and Load Databases with Password Protection

You can protect your database dump from unauthorized loads using the `password` parameter of the `dump database` command. If you include the `password` parameter when you make a database dump, you must also include this password when you load the database.

The partial syntax for the password-protected `dump database` and `load database` commands are:

```
dump database <database_name> to <file_name> [ with passwd = <password> ]
```

```
load database <database_name> from <file_name> [ with passwd = <password> ]
```

where:

- `<database_name>` – is the name of the database that is being dump or loaded.
- `<file_name>` – is the name of the dump file.
- `<password>` – is the password you provide to protect the dump file from unauthorized users.

Your password must be between 6 and 30 characters long. If you provide a password that is less than 6 or greater than 30 characters, an error message is issued. If you provide an incorrect password when you attempt to load the database, an error message is issued and the command fails.

For example, the following uses the password "bluesky" to protect the database dump of the `pubs2` database:

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

The database dump must be loaded using the same password:

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

10.6.1 Passwords and Character Sets

You can load the dump only to another server with the same character set. For example, if you attempt to load a dump from a server that uses an ASCII character set to a server that uses a non-ASCII character set, the load fails because the value of the ASCII password is different from the non-ASCII password.

Passwords entered by users are converted to the local character set. Because ASCII characters generally have the same value representation across character sets, if a user's password is in an ASCII character set, the passwords for `dump` and `load` are recognized across all character sets.

10.7 Residual Data Removal

To strengthen the security of database data, remove residual data.

Some database operations that delete space do not always physically erase the data. This can pose a security threat, as this residual data may be visible to a user using the `dbcc` utility. You can automatically zero out residual data when users perform such database operations.

You cannot remove residual data from system databases such as `master`, `sybsystemdb`, and `sybsystemprocs`, or from system tables present in each database.

> **i** Note
>
> Deallocated space is unavailable for reuse until SAP ASE finishes zeroing out the residual data. Depending on the amount of data that SAP ASE has to remove, this might take a long time.

10.7.1 Operations That Can Result in Residual Data

Any operation that results in space deallocation can cause residual data.

Table 30: Operations That Can Result in Residual Data

Operation	Description
`drop table` and `drop index`	Dropping a table or index can result in the deallocation of extents, which can then be directly reused for other objects, allowing a user who runs `dbcc` to see the contents of a deleted table. Residual data is removed if you set the `erase residual data` option at the session, database, or table level.
`alter table`	`alter table` commands such as `alter table encryption` delete the physical copy of existing data and store it in encrypted form, but do not delete old ASCII (or clear text) data pages; these can be viewed by directly looking at the database devices or by using the `dbcc page` command. Residual data is removed if you set the `erase residual data` option at the session, database, or table level.
Worktable deallocation	If the user query involves aggregations on the columns, there might be column data in the worktables. These worktables process the user query, and are deallocated at the end of query execution. These worktables are deallocated while closing the table, and the task deallocates the pages. You can remove the residual data by setting/enabling the `erase residual data` option at the session/database level.
Sort page deallocation	`sort` allocates pages from disk for merging the internal runs of internal list merge sorts. The extents used for external merge are cached in sort metadata structures and are reused to write the final result of the sort. If the external merge allocates more extents than the size of the table, the cached extents are deallocated at the end of the sort operation, and should be zeroed out.
Page deallocation after a page becomes empty	A delete query on a table can result in page deallocation; that is, if all the rows of a page are deleted, the page is then deallocated, and such a page has residual data.
Dropping or shrinking a database	Physical data can remain in dropped pages after a drop/shrink database operation, but this data is not visible via a database utility. The only way to reuse this space is by running `create database` or `alter database`, after which the pages are initialized with zero.
`reorg rebuild`	Replicates the data into another space, then re-creates the indexes on the table and organizes the data. `reorg rebuild` then deallocates the old data (using the data it replicated). If the table/session/database-level option is not set and the table contains sensitive information, the old data still has residual data.

Operation	Description
`reorg compact`	Deallocates the page if it finds that all the rows on the page are committed deletes. This happens if you run `reorg compact` before the SAP ASE mechanism for automatic garbage collection executes. If the pages that are queued for `reorg compact` are from the tables that have the option `erase residual data` turned on, the residual data in such pages should be removed.
Deleting a temporary table	Since a temporary table is similar to any other table, dropping a temp table that contains sensitive data results in the removal of residual data. For temp tables, set the removal of residual data at the session or table level; the database-level option does not work, as these tables reside in `tempdb`, for which the database option cannot be set.

10.7.2 Enabling the Removal of Residual Data

You can mark data as sensitive, and configure SAP ASE to erase residual data after performing delete or update operations.

Table 31: Enabling the Removal of Residual Data at Different Levels

Level	Description
Database level	Use: `sp_dboption <dbname>, "erase residual data", true` When you set `erase residual data` to true, SAP ASE automatically removes all residual data after any operation resulting in deallocation. **i Note** Using `sp_dboption` does not allow for granular control, and may impact performance. If possible, consider enabling the feature at the session level by using the `set` command instead. By default, when you enable the option at a database level, all the tables you create are set to erase residual data unless you explicitly turn it off. To set the database-level option using `sp_dboption`, the user must be a system administrator or database owner.

Level	Description
Session level	Use: `set erase_residual_data {on | off}` Using `set` allows you to enable or disable the removal of residual data based on your needs. When you enable the option at a session level, all page deallocations during that session have their residual data removed. This includes page deallocations of tables that have the `erase residual data` option turned off explicitly. This option can be set by any user for the particular session; no special permissions are required.

Table level	To create a new table, use:
	`create table <table_name> with "erase residual data" {on | off}`
	By default, nothing is set or enabled, which means that the feature is off for this table.
	To change the settings of an existing table to enable or disable the removal of residual data, use:
	`alter table <table_name> set "erase residual data" {on | off}`
	When you set the `erase residual data` option for a table, SAP ASE cleans up deallocated space after any operation on the table, such as `drop table`, `delete row` (which causes page deallocation), `alter table`, or `drop index`, that results in residual data.
	When you set the option for a table to on/off explicitly, you enable/disable data erasure for any operation that leaves residual data in the table respectively, regardless of its database settings.
	To set the table-level option using `create table` or `alter table`, the user must be listed in the `sysusers` table for that database and should have `create table` permissions. For `alter table`, the user must be a table owner or the database owner (who can impersonate as table owner using the `setuser` command) to be able to set this option.

10.7.3 Examples of Residual Data Removal

Examples show the removal of residual data.

The examples use these two tables:

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

Example: Example 1

Residual data is automatically removed for `t1` since it has the `erase residual data` option on, but not for `t2`, since it does not have the `erase residual data` option set, and the `dboption`/session-level option is not set:

```
drop table t1
go
drop table t2
go
```

Example: Example 2

Residual data is removed for both `t1` and `t2`, because `erase_residual_data` is set at the session level, which overrides the table-level option that was set to explicitly turn `erase_residual_data` off:

```
set erase_residual_data on
go
drop table t1
go
drop table t2
go
set erase_residual_data off
go
```

Example: Example 3

The option to erase residual data is turned on for table `t1` because it is set at the database level, so that both the `drop table` and `truncate table` commands for `t1` result in the cleanup of all residual data from its pages.

Table `t2`, however, has the `erase residual data` option turned off explicitly, as it was created with the `erase residual data off` clause. For this reason, residual data is not removed, even though the `erase residual data` option is set to true at the database level. As a result, residual data remains, even after running `drop table` and `truncate table` on `t2`:

```
create database db1
```

```
go
sp_dboption db1, "erase residual data", true
go
use db1
go
create table t1 (col int)
go
insert t1 values ...
go
create table t2 (col1 int, col2 char(10)) with "erase residual data" off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
drop table t2
go
```

Example: Example 4

In this example:

- Table `t1` does not have `erase residual data off` set explicitly, but does have it set at the database level, resulting in the removal of residual data from `t1` when you run `truncate table t1`.
- Table `t2` has the `erase residual data` option set at the time it was created because the option was set at the database level. This results in the removal of residual data from `t2` when you run `truncate table t2`.
- Table `t3` is marked with `erase residual data off` explicitly, so that even though `sp_dboption` sets `erase residual data` to true, residual data is not removed when SAP ASE runs `truncate table t3`.

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data" off
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

Example: Example 5

In this example,

- Although both `t1` and `t2` tables had the `erase residual data` option not set by default, because `erase_residual_data` was turned on at the session level just before the `truncate table` command was executed, the residual data is removedon both `t1` and `t2`
- Although table `t3` has the `erase residual data` option explicitly set to off, residual data is still removed when the `truncate` command is executed because the `erase_residual_data` option is set at session level.

```
create database db1
go
use db1
go
create table t1(col int)
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data" off
go
set erase_residual_data on
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

10.7.4 Combinations of Settings for Removing Residual Data

SAP ASE removes residual data depending on what the settings are at the various levels.

Table 32: Combinations of Settings and Residual Data Removal Behavior

Table Level	Session Level	Database Level	Cleanup
Off	Off	On	Yes
Off	On	Off	Yes
On	Off	Off	Yes
On	Off	On	Yes
On	On	Off	Yes
Off	On	On	Yes
Off, set explicitly	On	Off	Yes
Off, set explicitly	On	On	Yes
Off, set explicitly	Off	On	No

Of these settings, note these two combinations in particular:

- Combination 1:

Table Level	Session Level	Database Level	Clean-up
Off, set explicitly	On	Off	Yes

When the session-level option is on, the removal of residual data for the deallocated pages of the table happens even when the table-level option for `erase residual data` is explicitly set to off.
For this reason, enable the session-level option only if the current session involves sensitive data that requires removal, as the tables involved in the session that do not possess this property also remove residual data, causing a performance degradation.

- Combination 2:

Table Level	Session Level	Database Level	Clean-up
Off, set explicitly	Off	On	No

Even though the database-level option is on, the removal of residual data for the deallocated pages of the table is not performed, because the table-level option is explicitly turned off.
For this reason, if a table contains sensitive data, do not turn the table-level option explicitly off, as the deallocated pages contain residual data even if the database-level option is turned on.

10.7.5 Check Whether Residual Data Removal Is Enabled

Use system procedures to see whether you have enabled the removal of residual data at the table, database, and session levels.

Checking at the Table Level

Use `sp_help` to check whether the `erase residual data` option is set at the table level. For example:

```
alter table t1 set "erase residual data" on
go
sp_help t1
go
Name    Owner   Object_type    Object_status
    Create_date
------ -------- -------------- --------------------------
    -----------------------------------
t1      dbo     user table     keep first text page erase residual data
    Jul 30 2013 10:30PM
(1 row affected)
```

Checking at the Database Level

Use `sp_helpdb` to check whether the `erase residual data` option is set at the database level. For example:

```
use master
go
sp_dboption db1, "erase residual data", true
go
sp_helpdb db1
```

```
go
name   db_size      owner   dbid     created         durability
     lobcomplvl   inrowlen      status
------ ------------ ------- -------- -------------- ----------
     ------------ -----------   ------------------------------
db1     700.0 MB    sa        4      Jul 22, 2013    full
     0             NULL         erase residual data
(1 row affected)
```

Checking at the Session Level

Use `sp_show_options` to check whether the `erase residual data` option is set at the session level. For example:

```
set erase_residual_data on
go
sp_show_options
go
number     name
------     -----------------------------------
    […]
    110    erase_residual_data
(13 rows affected)
(return status = 0)
```

10.7.6 Limitations

There are some objects from which you cannot remove residual data.

Table 33: Limitations and Restrictions for Removing Residual Data

Limitation	Description
Page-level deallocation cleanups	You **can** remove residual data at the page deallocation level as long as you enable the feature. When you do, however, residual data is removed only for the page deallocation itself; residual data is not removed as a result of any row-level deletion that does not cause page deallocation.
Log records	Residual data is not removed from `syslogs` after deallocation, and log records are not cleaned up.
In-memory databases	You cannot remove residual data from pages that belong to an in-memory database.
Proxy tables	You cannot remove residual data from proxy tables.
System databases	You cannot remove residual data from system databases such as `master`, system tempdb, and `sybsystemprocs`. In addition, residual data is not removed at the database level when you perform deallocation operations that involve tables created in `systemtempdb`.

Limitation	Description
Worktables and sort pages	If any operation creates worktables or sort pages, those pages are deallocated if you set the database- or session-level option to remove residual data.
Dropping databases	A fully durable database with the `erase residual data` option on may experience data loss–that is, data recovery cannot be fully ensured–if the drop database operation is aborted or rolled back for any reason.

11 Auditing

A principal element of a secure system is accountability. One way to ensure accountability is to audit events on the system. Many events that occur in SAP ASE can be recorded.

Auditing is an important part of security in a database management system. An audit trail can be used to detect penetration of the system and misuse of resources. By examining the audit trail, a system security officer can inspect patterns of access to objects in databases and can monitor the activity of specific users. Audit records are traceable to specific users, which may act as a deterrent to users who are misusing the system.

Each audit record can log the nature of the event, the date and time, the user responsible for it, and the success or failure of the event. Among the events that can be audited are log ins and log outs, server starts, use of data access commands, attempts to access particular objects, and a particular user's actions. The audit trail, or log of audit records, allows the system security officer to reconstruct events that occurred on the system and evaluate their impact.

The system security officer is the only user who can start and stop auditing, set up auditing options, and process the audit data. As a system security officer, you can establish auditing for events such as:

- Server-wide, security-relevant events
- Creating, deleting, and modifying database objects
- All actions by a particular user or all actions by users with a particular role active
- Granting or revoking database access
- Importing or exporting data
- Log ins and log outs

> **i** Note
>
> Permission requirements for operations mentioned in this book assume that granular permissions is disabled. Operations may differ when granular permissions is enabled.

Related Information

Manage User Permissions [page 167]

11.1 The Audit System

The audit system consists of the `sybsecurity` database, an audit queue, configuration parameters, and system procedures.

- The `sybsecurity` database – contains global auditing options and the audit trail

- The in-memory audit queue – to which audit records are sent to the audit queue before they are written to the audit trail
- Configuration parameters – used for managing auditing
- System procedures – used for managing auditing

11.1.1 The sybsecurity Database

The `sybsecurity` database is created during the auditing installation process. In addition to all the system tables found in the `model` database, it contains `sysauditoptions`, a system table for keeping track of server-wide auditing options, and system tables for the audit trail.

`sysauditoptions` contains the current setting of global auditing options, such as whether auditing is enabled for disk commands, remote procedure calls, ad hoc user-defined auditing records, or all security-relevant events. These options affect the entire SAP ASE server.

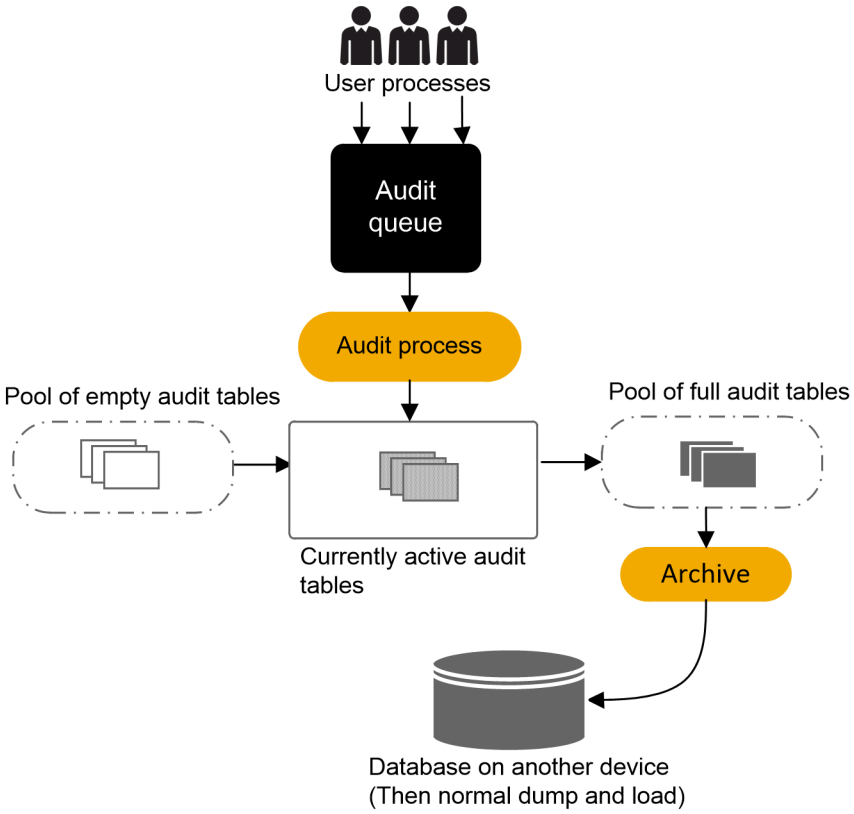
11.1.1.1 The Audit Trail

SAP ASE stores the audit trail in system tables named `sysaudits_01` through `sysaudits_08`.

When you install auditing, you determine the number of audit tables for your installation. For example, if you choose to have two audit tables, they are named `sysaudits_01` and `sysaudits_02`. At any given time, only one audit table is current. SAP ASE writes all audit data to the current audit table. A system security officer can use `sp_configure` to set (or change) which audit table is current.

SAP recommends two or more audit tables, with each table on a separate audit device. This allows you to set up a smoothly running auditing process in which audit tables are archived and processed with no loss of audit records and no manual intervention.

> ⚠ Caution
>
> SAP strongly recommends against using a single audit table on production systems. If you use only a single audit table, you may lose audit records.



The auditing system writes audit records from the in-memory audit queue to the current audit table. When the current audit table is nearly full, a threshold procedure can automatically archive the table to another database. The archive database can be backed up and restored with the `dump` and `load` commands. Use archive database access for read-only access to archived audit tables from backup. See *Archive Database Access*, in the *System Administration Guide, Volume 2*.

Related Information

Single-Table Auditing [page 377]
Audit Trail Management [page 368]

11.1.2 The Audit Queue

When an audited event occurs, an audit record first goes to the in-memory audit queue. The record remains in memory until the audit process writes it to the audit trail.

You can configure the size of the audit queue with the `audit queue size` parameter of `sp_configure`.

Before you configure the size of the audit queue, consider the trade-off between the risk of losing records in the queue if the system crashes and the loss of performance when the queue is full. As long as an audit record is in the queue, it can be lost if the system crashes. However, if the queue repeatedly becomes full, overall system

performance is affected. If the audit queue is full when a user process tries to generate an audit record, the process sleeps until space in the queue becomes available.

> **i** Note
>
> Because audit records are not written directly to the audit trail, you cannot count on an audit record being stored immediately in the current audit table.

11.2 Manage the Audit System

Task for managing the audit system.

1. Install auditing – set the number of audit tables and assign devices for the audit trail and the `syslogs` transaction log in the `sybsecurity` database.
2. Set up audit trail management – write and establish a threshold procedure that receives control when the current audit table is nearly full. The procedure automatically switches to a new audit table and archives the contents of the current table. In addition, this step involves setting the `audit queue size` and the `suspend audit when device full` configuration parameters.
3. Set up transaction log management in the `sybsecurity` database – determine how to handle the `syslogs` transaction log in the `sybsecurity` database, how to set the `trunc log on chkpt` database option and establishing a last-chance threshold procedure for `syslogs` if `trunc log on chkpt` is off.
4. Set auditing options – use `sp_audit` to establish the events to be audited.
5. Enable auditing – use `sp_configure` to turn on the auditing configuration parameter. SAP ASE begins writing audit records to the current audit table.
6. Restart auditing – use `sp_audit restart` to restart auditing if the audit process is forced to terminate due to an error.

11.2.1 Preinstallation Recommendations for Auditing Devices

Complete the preinstallation recommendations before installing auditing.

- Determine the location of the devices for the `sybsecurity`, `syslogs`, and `sysaudits` table devices. You will need to provide this information later.
- Configure your system with the minimum number of auditing devices you require – you must configure at least three devices. You can use `sp_addaudittable` to add more auditing devices later. See the *Reference Manual: Procedures*.
- Install auditing tables and devices in a one-to-one ratio. Tables that share the same device also share the same upper threshold limit. These tables cannot be used sequentially when a device fills up, because they both reside on the same device.
- Install each auditing table on its own device. This enables you to set up a smoothly running auditing system with no loss of auditing records. With two auditing tables, when one fills up, you can switch to the other. With a third auditing table, if one device fails, the system security officer can install a new threshold procedure that changes the device rotation to skip the broken device until the device is repaired.

- Make the device larger than the table. When you use only three auditing tables and devices, the size of the table and the size of the device can be similar, because you can obtain more auditing capacity by adding more auditing tables and devices. When you are working toward the upper table and device limit (six to eight), you may want to make the device considerably larger than the table. You can later expand the table size toward the upper size of the device when a larger auditing capacity is desired, and few or no device additions are available.

If you are using a file system device, either set the `dsync` attribute on, or use the `directio` attribute with that device.

11.2.2 Install the Audit System

There are two methods for installing auditing for the first time in SAP ASE.

You can use the `auditinit` utility or an install script to install the audit system. The utilities and install scripts are located here:

- On UNIX, use one of the following:
  - `$SYBASE/$SYBASE_ASE/bin/auditinit`
  - `$SYBASE/$SYBASE_ASE/scripts/installsecurity`
- On Windows, use one of the following:
  - `%SYBASE%\%SYBASE_ASE%\bin\auditinit.exe`
  - `%SYBASE%\%SYBASE_ASE%\scripts\instsecu`

> **i** Note
>
> After auditing is installed, no auditing occurs until a System Security Officer enables auditing using `sp_configure`.

11.2.2.1 Install Auditing on UNIX with installsecurity

Install the SAP ASE auditing system on the UNIX platform using the `installsecurity` script. After you complete this step, you can enable auditing.

Context

The `$SYBASE/ASE-16_0/scripts` directory contains the `installsecurity` script for installing auditing.

> **i** Note
>
> Select your device size for data and log depending on the logical page size. This example assumes a server that uses a logical page size of 2K. Fore more information about device sizes, see the *Configuration Guide for UNIX > SAP ASE Devices and System Databases > Determine the Size of a Database Device*.

Procedure

1. Determine an available device number to use for the auditing device.
2. Create the auditing devices and auditing database with the `disk init` and `create database` commands. For example:

```
disk init name = "auditdev01",
    physname = "/dev/dsk/auditdev",
    size = "10M"
```

```
disk init name = "auditlogdev01",
    physname = "/dev/dsk/auditlog",
    size = "2M"
```

```
create database sybsecurity on auditdev01 log on auditlogdev01
```

3. Use `isql` to execute the `installsecurity` script:

```
cd $SYBASE/ASE-16_0/scripts
setenv DSQUERY <server_name>
isql -Usa -P<password> -S<server_name> -iinstallsecurity
```

4. Shut down and restart the SAP ASE server.

Results

When you have completed these steps, the `sybsecurity` database has one audit table (`sysaudits_01`) created on its own segment. You can enable auditing at this time, but should add more auditing tables with `sp_addauditttable`. For information about `disk init`, `create database`, and `sp_addauditttable`, see *Reference Manual: Procedures* and *Reference Manual: Commands*.

11.2.2.2 Install Auditing on Windows with instsecu

Install the SAP ASE auditing system on the Windows platform using the `instsecu` script. After you complete this step, you can enable auditing.

Context

The `%SYBASE%\ASE-16_0\scripts` directory contains the `instsecu` script for installing auditing.

Procedure

1. Determine the next available device number to use for the auditing device.
2. Open a Command Prompt window.
3. Start the `isql` program as user “sa”:

   ```
   isql -Usa -P<password> -S<server_name>
   ```

4. At the `isql` prompt, use the `disk init` command to create the auditing devices.

   An example of configuring the auditing database:

   ```
   declare @devno int
   select @devno = max(low/16777216)+1 from sysdevices
   disk init
   name = "auditdev",
   physname = "%SYBASE%\data\sybaud.dat",
   vdevno = @devno,
   size = 5120
   ```

   An example of configuring the auditing database log:

   ```
   declare @devno int
   select @devno = max(low/16777216)+1 from sysdevices
   disk init
   name = "auditlogdev",
   physname = "%SYBASE%\data\sybaudlg.dat",
   vdevno = @devno,
   size = 1024
   ```

5. Create the auditing database:

   ```
   create database sybsecurity on auditdev
   log on auditlogdev
   ```

6. Exit `isql`:

   ```
   exit
   ```

7. Change to the `scripts` directory:

   ```
   cd %SYBASE%\ASE-16_0\scripts
   ```

8. Set the DSQUERY environment variable:

   ```
   set DSQUERY = <server_name>
   ```

9. Start the `isql` program as user “sa” with the `instsecu` script as the input file:

   ```
   isql -Usa -P<password> -S<server_name > -iinstsecu
   ```

10. Shut down and restart SAP ASE.

Results

Once you have completed these steps, the `sybsecurity` database has one audit table (`sysaudits_01`) created on its own segment. Add additional auditing tables with `sp_addaudittable`. No auditing occurs until a system security officer enables auditing with the auditing system procedures. For information about `disk init`, `create database`, and `sp_addaudittable`, see the *Reference Manual: Procedures* and *Reference Manual: Commands*.

11.2.2.3 Install Auditing with auditinit (Windows and UNIX)

Install auditing with the `auditinit` utility.

The `auditinit` process of installing an audit system involves:

- Configuring the server
- Creating devices for the audit table
- Creating a device for the audit database transaction log

The `auditinit` utility is located here:

- UNIX: `$SYBASE/ASE-16_0/bin/auditinit`
- Windows: `%SYBASE%\ASE-16_0\bin\auditinit.exe`

For more information about the `auditinit` utility, see the *Utility Guide*.

11.2.2.3.1 Configure a Server with auditinit

Configure the SAP ASE server for auditing with the `auditinit` utility.

Procedure

1. Do one of the following:
   - (On UNIX) Source the `SYBASE.csh` or `SYBASE.sh` environment variable file.
   - (On Windows) Go to the directory `%SYBASE%\ASE-16_0\bin`
2. Execute `auditinit`.

   The following menu is displayed:

   ```
   AUDITINIT
   1. Release directory:  /usr/u/sybase
   2. Configure a Server product
   ```

3. Select `Configure a Server product`.

   As a result, a list of available products is displayed.

4. From the product list, select `Adaptive Server`, or on Windows, `SQL Server`.
5. Select `Configure an existing SAP Server`.
6. Select the server to configure.
7. Provide the sa password for the server that you selected.
8. Type *Ctrl+a* and Enter to accept the sa account values.

   As you proceed through the menus in `auditinit`, you can change any default values that appear. As you finish each menu, type *Ctrl+a* to accept the defaults or changed values and move to the next menu.
9. From the `Sybase Server Configuration` screen, select `Configure Auditing`.

   The following menu is displayed:

```
CONFIGURE AUDITING
1.  Configure auditing:  no
2.  create new device for the sybsecurity database:  no
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
```

   Type *Ctrl+f* to scroll forward.
10. From the `Configure Auditing` screen, select `Configure Auditing`.

    As a result, `auditinit` re-displays the `Configure Auditing` menu with the value “yes” displayed for `Configure Auditing`.

```
CONFIGURE AUDITING
1.  Configure auditing:  no
2.  create new device for the sybsecurity database:  yes
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
```

Next Steps

To continue with the audit installation, you must add devices for the audit tables and then a device for the audit database transaction log.

Related Information

Create a Device for an Audit Table with auditinit [page 343]
Create a Device for the Audit Transaction Log with auditinit [page 345]

11.2.2.3.2 Create a Device for an Audit Table with auditinit

Create audit table devices using `auditinit`.

Prerequisites

You must start the `auditinit` utility and configure the server before adding devices for an audit table See *Configure a Server with auditinit*.

Procedure

1. When the server is selected to be configured, the following `auditinit` menu is displayed:

   ```
   CONFIGURE AUDITING
   1.  Configure auditing:  yes
   2.  create new device for the sybsecurity database:  no
   3.  Add a device for audit table(s)
   4.  Add a device for the audit database transaction log
   5.  Delete a device entry
   6.  Change a device entry

   List of devices for the audit tables:
   Logical name    Physical name    Segment name    Table name    Size
   Device for the audit database transaction log:
   Logical name    Physical name    Segment name    Table name    Size
   ```

   Press Ctlr+f to scroll forward.
2. From the same screen, select `Add a device for sybsecurity database`.

   As a result, `auditinit` re-displays the `Configure Auditing` menu with the value "yes" displayed for `Add a device for sybsecurity database`. Type *Ctrl+f* to scroll forward.
3. From the `Configure Auditing` screen, select `Add a Device for Audit Table(s)`.

   `auditinit` displays the following menu:

   ```
   ADD/CHANGE A NEW DEVICE FOR AUDITING
   1. sybsecurity physical device name:
   2. Logical name of the device:
   3. Size of the device (Meg):
   4. Device size for auditing:
   ```
4. Select `sybsecurity physical device name`.
5. Enter the full path of the physical device (file system or raw partition).

   ```
   Enter the physical name of the device to use for the audit database (default
   is " "):
   /dev/<path_to_partition>
   ```
6. Type *Enter* key to acknowledge the warning.

`auditinit` re-displays the `Add/Change a New Device for Auditing` menu, which displays the physical name of the device:

```
ADD/CHANGE A NEW DEVICE FOR AUDITING
1. sybsecurity physical device name:  /secret1/sybase_dr/install/aud1.dat
2. Logical name of the device:
3. Size of the device:
4. Device size for auditing:
```

Type *Ctlr+f* to scroll forward.

7. Proceed through the remaining items on this menu.

> **i Note**
>
> The `Size of the device` value must be equal to or greater than the `Device size for auditing` value.If you are following SAP auditing guidelines, you need not change the value displayed in `Device size for auditing`.

8. Type *Ctrl+a* to accept the settings. `auditinit` returns to the `Configure Auditing` menu and displays the device you have created.

```
CONFIGURE AUDITING
1.  Configure auditing:  yes
2.  create new device for the sybsecurity database:  yes
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
List of devices for the audit tables:
Logical name   Physical name    Segment name         Table name    Size
7.Audit_01' secret1/sybase_dr/install/aud1.dat’ sysaudits_01 5
```

9. To add multiple audit devices, repeat steps 1– 6. You can add as many as eight devices. SAP recommends adding three or more audit table devices.
   After adding a device, `auditinit` returns to the `Configure Auditing` menu and displays all the devices you have created.

```
CONFIGURE AUDITING
1.  Configure auditing:  yes
2.  create new device for the sybsecurity database:  yes
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
List of devices for the audit tables:
Logical name    Physical name    Segment name         Table name     Size
7. Audit_01'   /secret1/sybase_dr/install/aud1.dat’ sysaudits_01   5
8. Audit_02'   /secret1/sybase_dr/install/aud2.dat' sysaudits_02   5
```

Next Steps

To continue with the audit installation, you must add a device for the audit database transaction log.

Related Information

Create a Device for the Audit Transaction Log with auditinit [page 345]

11.2.2.3.3 Create a Device for the Audit Transaction Log with auditinit

Create a device for the audit transaction log using `auditinit`.

Prerequisites

You must start the `auditinit` utility, configure the server, and add devices for an audit table before adding a transaction log device. See *Configure a Server with auditinit* and *Create a Device for an Audit Table with auditinit*.

Procedure

1. When the audit table devices have been created, the following `auditinit` menu is displayed:

```
CONFIGURE AUDITING
1.  Configure auditing:  yes
2.  create new device for the sybsecurity database:  yes
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
List of devices for the audit tables:
Logical name    Physical name    Segment name         Table name     Size
7. Audit_01'   /secret1/sybase_dr/install/aud1.dat' sysaudits_01   5
8. Audit_02'   /secret1/sybase_dr/install/aud2.dat' sysaudits_02   5
```

   Type *Ctlr+f* to scroll forward.

2. From the `Configure Auditing` menu, select `Add a device for the audit database transaction log`.

   `auditinit` displays the `Add/Change a New Device for Auditing` menu.

```
ADD/CHANGE A NEW DEVICE FOR AUDITING
1.  sybsecurity physical device name:
2.  Logical name of the device:
3.  Size of the new device (Meg):
4.  Device size for auditing:
```

3. Select `sybsecurity physical device name`.

   `auditinit` prompts for the physical name and supplies you with a default, if available:

```
Enter the physical name of the device to use for the sybsecurity database
(default is''):
```

```
/dev/<path_to_partition>
```

where <path_to_partition >is the path to the raw partition for the device.

4. Enter the full path name of a physical device.
5. Type *Enter*.

   auditinit displays the Add/Change a New Device for Auditing menu and the value you selected for the physical name of the device.

```
ADD/CHANGE A NEW DEVICE FOR AUDITING
1.sybsecurity physical device name:
      /secret1/sybase_dr/install/auditlog.dat
2.Logical name of the device:
3.Size of the device:
4.Device size for auditing:
```

   Type *Ctlr+f* to scroll forward.

6. Proceed through the remaining items on this menu. As you do so, be aware of:
   - SAP recommends a minimum size of 2MB for the size of the transaction log.
   - auditinit displays the size in Size of the device and in Device size for auditing in the Add/Change a New Device for Auditing menu.
   - The Device size for auditing default value is equal to Size of the device, based on the assumption that you may want to devote the entire device to log for the auditing task. To use only a subset of the device, edit the Size of the device value.
7. Type *Ctrl+a* to accept the settings that appear in the Add/Change a New Device for Auditing menu.

   auditinit returns to the Configure Auditing menu and displays all the devices you have created.

```
CONFIGURE AUDITING
1.  Configure auditing:  yes
2.  create new device for the sybsecurity database:  yes
3.  Add a device for audit table(s)
4.  Add a device for the audit database transaction log
5.  Delete a device entry
6.  Change a device entry
List of devices for the audit tables:
Logical name    Physical name    Segment name         Table name     Size
7. Audit_01'   /secret1/sybase_  dr/install/aud1.dat' sysaudits_01   5
8. Audit_02'   /secret1/sybase_  dr/install/aud2.dat' sysaudits_02   5
9. auditlog    /secret1/.../auditlog.dat logsegment   syslogs        2
```

8. When you are ready to execute the audit configuration, type *Ctrl+a*.
9. On the SAP Server Configuration screen, type *Ctrl+a* again. You see:

```
Execute the SAP Server Configuration now?
```

10. Enter “y” (yes).

    auditinitexecutes the tasks to install auditing. When installation completes successfully, you see:

```
Running task: install auditing capabilities.
.....................Done
Auditing capability installed.
Task succeeded: install auditing capabilities.
Configuration completed successfully.
Press <return> to continue.
```

Related Information

Configure a Server with auditinit [page 341]
Create a Device for an Audit Table with auditinit [page 343]

11.2.3 Enable or Disable Auditing

After auditing is installed, no auditing occurs until a System Security Officer enables auditing using `sp_configure`.

> **i Note**
>
> Before enabling auditing with `sp_configure`, restart the server.

Use `sp_configure` with the `auditing` configuration parameter to enable or disable auditing.

The syntax is:

```
sp_configure "auditing", [0 | 1 ]
```

- `1` – enables auditing.
- `0` – disables auditing.

For example, to enable auditing, enter:

```
sp_configure "auditing", 1
```

> **i Note**
>
> When you enable or disable auditing, SAP ASE automatically generates an audit record.

11.2.4 Restart Auditing

If the audit process is forced to terminate due to an error, it can be manually restarted.

Context

Enter:

```
sp_audit restart
```

The audit process can be restarted provided that no audit was currently running, but the audit process must be enabled with `sp_configure "auditing", 1`.

11.2.5 Configure the Audit System

Configure the auditing system using `sp_configure` and system procedures.

Use the `sp_configure` auditing parameters to manage the auditing process:

- `auditing` – enables or disables auditing for the entire SAP ASE server. The parameter takes effect immediately upon execution of `sp_configure`. Auditing occurs only when this parameter is enabled.
- `audit queue size` – establishes the size of the audit queue. Because the parameter affects memory allocation, the parameter does not take effect until the SAP ASE server is restarted.
- `suspend audit when device full` – controls the behavior of the audit process when an audit device becomes full. The parameter takes effect immediately upon execution of `sp_configure`.
- `current audit table` – sets the current audit table. The parameter takes effect immediately upon execution of `sp_configure`.

Use these system procedures to manage the auditing process:

- `sp_audit` – enables and disables auditing options. This is the only system procedure required to establish the events to be audited.
- `sp_displayaudit` – displays the active auditing options.
- `sp_addauditrecord` – adds user-defined audit records (comments) into the audit trail. Users can add these records only if a system security officer enables ad hoc auditing with `sp_audit`.

11.2.5.1 Audit Configuration Changes

SAP also recommends that changes to SAP ASE configuration be audited.

There are two ways to enable audit of configuration changes made by `sp_configure`:

- use the audit option `exec_procedure` on the object of `sp_configure` to directly audit `sp_configure`,
- use the audit option `security` which includes configuration changes made by `sp_configure` and other procedures.

To capture configuration changes made to the configuration file directly, use operating system provided file system auditing on the configuration files.

Follow the recommendations of your operating system vendor to enable auditing of configuration files.

For example:

- On Linux 2.6 kernel, use the `auditd(8)` daemon and related utilities `auditctl`, `ausearch`, and `aureport`.
- On AIX, the audit system is configured in `/etc/security/audit/config` and started using the audit `start` command. The IBM Redbook "Accounting and Auditing on AIX 5L" can be referenced for AIX systems.
- On Windows, auditing on files and folders is enabled by setting properties in the Security tab. For example, right-click the file or folder, then select *Properties* > *Security* > *Advanced* > *Auditing* and follow the dialog box. More information for Windows can be found on the Microsoft support site.

11.2.5.2 Correlate SAP ASE and Operating System Audit Records

The easiest way to link SAP ASE audit records with operating system records is to make SAP ASE login names the same as operating system login names.

Alternatively, the system security officer can map users' operating system login names to their SAP ASE login names. However, this approach requires ongoing maintenance, as login names for new users must be recorded manually.

11.2.5.3 Set Auditing Options

After you have installed auditing, use `sp_audit` to set auditing options.

The syntax for `sp_audit` is:

```
sp_audit <option>, <login_role_name>, <object_name> [,<setting>]
```

If you run `sp_audit` with no parameters, it provides a complete list of the options.

For details about `sp_audit`, see *Reference Manual: Procedures* > *sp_audit*.

> **i** Note
>
> Auditing does not occur until you activate auditing for the server.

Related Information

Enable or Disable Auditing [page 347]

11.2.5.3.1 Determine Current Auditing Settings

To determine the current auditing settings for a given option, use `sp_displayaudit`.

The syntax is:

```
sp_displayaudit ["procedure" | "object" | "login" | "database" | "global" |
    "default_object" | "default_procedure" | "role" [, "<name>"]]
```

For more information, see `sp_displayaudit` in *Reference Manual: Procedures*.

11.2.5.3.2 Full-Text Auditing

SAP ASE supports full-text auditing for DDL and DML commands, printing full command text in the second position of the `extrainfo` column when you enable relevant auditing options.

Full-text auditing also records parameters, printing them in the fifth position of the `extrainfo` column. Passwords, encryption key values, and encrypted column parameters are replaced with a fixed length string of asterisks in the audit records. Extra white space and comments are filtered to save space in the logs. Comments are included in the logs only if they are quoted such as: `"/* audit comment */"` or bracketed, such as `"[/* audit comment */]"`.

> **i** Note
>
> When DML commands are executed from stored procedures, more procedure cache is required to store command text and parameter values. SAP recommends that you increase the size of the audit log and procedure cache to account for the extra space requirements of full-text auditing.

For a complete list of commands supported by full text auditing, see Reading the extrainfo Column [page 379].

11.2.5.3.2.1 Examples of Full-Text Auditing

In each of the examples of full text auditing, enable auditing option, execute a command or procedure, and view auditing record in `sysaudits_01`.

Example 1

Enable `create` auditing option for the "master" database:

```
sp_audit "create", "all", "master", "on"
go
```

Create a table:

```
create table t1(c1 int,c2 int)
```

Select the event from `sysaudits_01`. The full text of create statement is displayed (in **bold**):

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 10
go
event  extrainfo
-----------------------------------------------------------------
10     sa_role sso_role oper_role sybase_ts_role mon_role; create table t1
   (c1 int,c2 int) ; ; ; ; ; sa/ase;
```

Example 2

Enable `dbcc` auditing option:

```
sp_audit "dbcc", "all", "all", "on"
go
```

Run `dbcc checktable` on t1:

```
dbcc checktable('master.dbo.t1')
go
```

The audit table contains the full text of the statement (in **bold**):

```
select event, extrainfo from sysaudits_01 where event = 81
go
event  extrainfo
------------------------------------------------------------------
81     sa_role sso_role oper_role sybase_ts_role mon_role; dbcc checktable
   ('master.dbo.t1'); ; ; ; ; sa/ase;
```

Example 3

Enable auditing for executing `drop` on "test1" database:

```
sp_audit "drop", "all", "test1", "on"
go
```

Create and then drop tables t1 and t2:

```
create table t1(c1 int, c2 int)
create table t2(c1 int, c2 int)
go
drop table t1,t2
go
```

The audit table contains two full text audit records for the `drop table` command. Since the command is executed on two different objects (t1 and t2), two separate audits records are generated, one for each object identified by an **objid** and **objname** :

```
select event, dbid, objid, objname, extrainfo from sybsecurity..sysaudits_01
    where event = 27
go
  event     objid        objname
  extrainfo
------------------------------------------------------------------
  27        992003534    t1
  sa_role sso_role oper_role sybase_ts_role mon_role; drop table t1, t2;
    ; ; ; ; sa/ase;
    27      1008003591   t2
  sa_role sso_role oper_role sybase_ts_role mon_role; drop table t1, t2;
    ; ; ; ; sa/ase;
```

Since `create` option has not been enabled for "test1" database, there are no audit records for this command.

Example 5

Enable `security_profile` auditing option:

```
sp_audit "security_profile", "all", "all", "on"
```

Run:

```
create login profile joe_lp
go
alter login profile joe_lp modify default database "sybsystemprocs"
go
drop login profile joe_lp
```

```
go
```

The audit table contains the full text of the statements (in **bold**):

```
select event,extrainfo from sybsecurity..sysaudits_01 where event in (137, 140,
141)
go
event extrainfo
-----------------------------------------
137 ; sa_role sso_role oper_role sybase_ts_role; create login profile joe_lp; ; ; ; ;
sa/ase;
140 ; sa_role sso_role oper_role sybase_ts_role; alter login profile joe_lp modify default
database "sybsystemprocs"; ; ; ; ; sa/ase;
141 ; sa_role sso_role oper_role sybase_ts_role; drop login profile joe_lp; ; ; ; ; sa/
ase;
```

For more examples, see Hide Sensitive Parameter Values [page 354], Print Parameter Values [page 352], and Filter Out Comments [page 357].

11.2.5.3.2.2 Print Parameter Values

In full-text auditing records, parameter names and values are printed in the fifth position of the `extrainfo` column.

Full-text auditing does not print parameter values for text, unitext and image datatypes. Instead, it prints the string: "`Text data`", "`Unitext data`", and "`Image data`" respectively in the result set. See example 5 below.

For a complete list of commands for which printing parameters is supported, see Reading the extrainfo Column [page 379].

Example 1

Enable `update` auditing option:

```
sp_audit "update", "all", "t1", "on"
```

And update table t1:

```
declare @val1 int
select @val1 = 1
update t1 set c1 = @val1
go
```

When you select the auditing event from `sysaudits_01`, SAP ASE displays full text of the `update` statement. In addition, parameters are displayed. The full text of the command and parameters are shown in **bold**:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 70
go
event  extrainfo
-----------------------------------------------------------------
70     sso_role oper_role sybase_ts_role mon_role; update t1 set c1 =
   @val1 ; ; ; @val1 = 1; ; sa/ase;
```

Example 2

Enable `disk` auditing option:

```
sp_audit "disk", "all", "all", "on"
```

And issue `disk resize` on test_db_dev:

```
declare @p_name varchar(50)
declare @p_dsize varchar(50)
select @p_name='test_db_dev'
select @p_dsize='32m'
disk resize name = @p_name, size = @p_dsize
```

When you select the auditing event from `sysaudits_01`, SAP ASE displays the full text of the `disk resize` statement and all parameters:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 100
go
event  extrainfo
-----------------------------------------------------------------------
100    sa_role sybase_ts_role; disk resize name = @p_name, size =
   @p_dsize; ; ; @p_name = test_db_dev, @p_dsize = 32m; ; user0/ase;
```

Example 3

Enable `select` auditing option for table t1:

```
sp_audit "select", "all", "t1", "on"
go
```

Create a stored procedure that selects from t1:

```
create proc p1 @val1 int, @val2 int
as
select * from t1 where c1 = @val1
and c2 = @val2
go
sp_audit 'select', 'all', 't1', 'on'
go
exec p1 10, 20
go
```

The audit table contains the full text of the `select` statement and parameters:

```
select event, extrainfo from sysaudits_01 where event = 62
go
event  extrainfo
-----------------------------------------------------------------------
62     sa_role sso_role oper_role sybase_ts_role mon_role; select * from t1
   where c1 = @val1 and c2 = @val2; ; ; @val1 = 10, @val2 = 20;
       ; sa/ase;
```

Example 4

Enable auditing for the `role` option:

```
sp_audit "role", "all", "all", "on"
```

Run:

```
declare @para1_role varchar(30)
```

```
declare @para2_grantee1 varchar(30)
select @para1_role='role1'
select @para2_grantee1 ='user4'
grant role @para1_role to @para2_grantee1
```

The audit table contains the full text of the statement and parameters:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 85
go
event  extrainfo
-----------------------------------------------------------------
85     sso_role; grant role @para1_role to @para2_grantee1; ; ;
       @para1_role = role1, @para2_grantee1 = user4; ; user1/ase;
```

Example 5

Full-text auditing does not print parameter values for text, unitext and image datatypes. Instead, it prints the string: "`Text data`", "`Unitext data`", and "`Image data`" respectively in the result set. In this example, "`Text data`" is printed:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 41
go
event  extrainfo
-----------------------------------------------------------------
41     sa_role sso_role oper_role sybase_ts_role mon_role; insert mytext
        values (10, @a); ; ; @a = Text data; ; sa/ase;
```

11.2.5.3.2.3 Hide Sensitive Parameter Values

In full-text auditing records, sensitive parameters such as passwords, encryption key values, and encrypted column parameter values are replaced with a fixed length string of asterisks.

Hide Passwords

Passwords are replaced with a fixed-length string of asterisks in the records contained in the audit logs.

Example 1

For example, if you enable `login_admin` audit option:

```
sp_audit "login_admin", "all", "all", "on"
```

And then execute:

```
alter login johnd with password 'oldpasswd' modify password 'newpasswd'
```

The command will appear in the audit log as:

```
select event, extrainfo from sysaudits_01 where event = 138
go
event  extrainfo
-----------------------------------------------------------------
138    sso_role oper_role sybase_ts_role mon_role; alter login johnd
```

```
        with password '******' modify password '******'; ; ; ;
        ; user1/ase;
```

This protects passwords from being seen by others with access to the audit log.

Example 2

Enable `role` and `password` audit options:

```
sp_audit "role", "all", "all", "on"
go
sp_audit "password", "all", "all", "on"
go
```

And alter the role "role1" by adding password:

```
alter role role1 add passwd "sap78475984ulkmf"
```

The audit table contains two records, one for "role" (event 85) ad one for "password" (event 134) option. Each record includes the full text of the command with obfuscated password (in **bold**):

```
select event, extrainfo from sysaudits_01 where event in (85, 134)
go
event  extrainfo
-------------------------------------------------------------------
 85   sa_role sso_role oper_role sybase_ts_role mon_role; alter role role1
   add passwd "******"; ; ; ; ; sa/ase;
134   sa_role sso_role oper_role sybase_ts_role mon_role; alter role role1
   add passwd "******"; ; ; ; ; sa/ase;
```

Example 3

Enable `dump` audit option:

```
sp_audit "dump" , "all", "tdb1", "on"
```

And execute `dump database` command::

```
create procedure dmpdb as
dump database tdb1 to "/tmp/tdb1.dmp" with passwd='sap12345'
go
exec dmpdb
go
```

When you select the auditing event from `sysaudits_01`, SAP ASE prints the full text of the command with obfuscated password:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 132
go
event  extrainfo
-------------------------------------------------------------------
70     sa_role sso_role oper_role sybase_ts_role mon_role; dump database
    tdb1 to "/tmp/tdb1.dmp" with passwd='******'; ; ; ; ; sa/ase;
```

Hide Encrypted Key Values

Encrypted key values are replaced with a fixed length string of asterisks in the audit records contained in the audit logs.

Example 1

Enable auditing for the `encryption_key` auditing option:

```
sp_audit 'encryption_key', 'all', 'tdb1', 'on'
```

Create an encryption key:

```
CREATE ENCRYPTION KEY loginA_key FOR AES WITH KEYLENGTH 128 PASSWD
0x0000a91c0c68fcada09e0
INIT_VECTOR RANDOM PAD NULL KEYVALUE
0x54db068edf1b934726a7537fb1e660387a6da06f5ff7df04f80192d7a84fc66301 KEYSTATUS 33
go
```

The audit table contains the full text of the statement with obfuscated encryption key value (in **bold**):

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 107
go
event extrainfo
------------------------------------------------------------
107  sso_role oper_role sybase_ts_role mon_role; CREATE ENCRYPTION KEY
      loginA_key FOR AES WITH KEYLENGTH 128 PASSWD****** INIT_VECTOR RANDOM PAD
      NULL KEYVALUE ****** KEYSTATUS 33; ; ; ; ; sa/ase;
```

> **i** Note
>
> When a sensitive parameter with a binary value is used in a stored procedure, its value is not obfuscated in the audit record.

Hide Encrypted Column Parameters

Encrypted column parameter values are replaced with a fixed length string of asterisks in the audit records contained in the audit logs.

Example 1

In this example, the values of parameters which are used to insert or update data to encrypted columns are obfuscated.

For example, the second column in table `t1` is encrypted:

```
declare @nr char(30)
declare @nr1 int
declare @nr2 char(30)
select @nr="aaa"
select @nr1=100
select @nr2="bbb"
insert t1 values(@nr, @nr1, @nr2)
go
```

The command appears in the audit log as:

```
select event, extrainfo from sysaudits_01 where event = 41
go
event  extrainfo
------------------------------------------------------------------
41     sa_role sso_role oper_role sybase_ts_role mon_role; insert t1 values
        (@nr, @nr1, @nr2); ; ; @nr = aaa, @nr1 = '******' , @nr2 = bbb;
        ; sa/ase;
```

11.2.5.3.2.4 Filter Out Comments

In full text auditing records, comments are filtered out unless they are quoted such as: "`/* audit comment */`" or bracketed, such as "`[/* audit comment */]`". Extra white space is filtered.

For example, if you enable auditing for executing `create` commands in "test1" database:

```
sp_audit "create", "all", "test1", "on"
```

And run:

```
set quoted_identifier on
go
create table "/* audit comment */" (col int)
create table /* audit comment */ "testtable" (col int)
go
```

The first audit record contains full text of the first `create table` command and the quoted comment. The second record includes full text of the `create table` command, but the comment is filtered out (since it was not quoted or bracketed):

```
select event,extrainfo from sybsecurity..sysaudits_01 where event = 10
go
event extrainfo
-----------------------------------------
10   sa_role sso_role oper_role sybase_ts_role mon_role; create table
   "/* audit comment */" (col int); ; ; ; ; sa/ase;
10   sa_role sso_role oper_role sybase_ts_role mon_role; create table
   "testtable" (col int); ; ; ; ; sa/ase;
```

The

11.2.5.3.3 Auditing Stored Procedures

In general, you can audit execution of stored procedures by issuing `sp_audit "exec_procedure"`, which generates an audit record containing the name of the stored procedure and the parameters.

Some system stored procedures can be audited after enabling specific audit options. See procedure-specific documentation in the reference manual.

11.2.5.3.3.1 Enable Auditing of Stored Procedures

Enable `exec_procedure` to audit execution of stored procedures. A majority of stored procedures can be audited using this option.

> **i** Note
>
> Before enabling `exec_procedure`, run `use sybsystemprocs`.

Enables auditing for a specific procedure:

```
sp_audit "exec_procedure", "all", <procedure_name>, "on"
```

Enables auditing for audit all new procedures:

```
sp_audit "exec_procedure", "all", "default procedure", "on"
```

Example

This example audits `sp_addmessage`:

```
use sybsystemprocs
sp_audit "exec_procedure", "all", "sp_addmessage", "on"
go
```

```
sp_addmessage 220000, "Test Message"
go
```

All the parameters entered for `sp_addmessage` are printed in position 5 of the `extrainfo` column:

```
select event, extrainfo from sybsecurity..sysaudits_01 where event = 38
go
event  extrainfo
------ -----------------------------------------------------
 38     sa_role sso_role oper_role sybase_ts_role mon_role; ; ; ; 220000,
      Test Message; ; ;
```

11.2.5.3.3.2 Enable Auditing for Authorization Checks

The audit option `sproc_auth` enables auditing for authorization checks that are performed inside system stored procedures.

Granular Permissions	Event
Enabled	146. The audit event 146 is only audited when the option `sproc_auth` is enabled.
Disabled	80. The audit event 80 is audited when the `sproc_auth` or `security` audit option is enabled.

Enables `sproc_auth`:

```
sp_audit "sproc_auth","all","all","on"
```

The permission needed to run the stored procedure is included in the fifth position of `extrainfo` column.

Example

Enable granular permissions:

```
sp_configure "enable granular permissions", 1
go
```

Enable `"sproc_auth"`:

```
sp_audit "sproc_auth","all","all","on"
```

Event 146 is recorded each time an authentication check is performed:

```
sp_audit "drop", "all", "testdb", "on"
go
select event, extrainfo from sybsecurity..sysaudits_01 where event = 146
go
event  extrainfo
---------------------------
146    sa_role sso_role oper_role sybase_ts_role mon_role; ; ; ;
        Permission:'manage auditing'; ; sa/ase;
```

Example

Disable granular permissions:

```
sp_configure "enable granular permissions", 0
go
```

(Skip this step if you performed it in example 1) Enable `"sproc_auth"`:

```
sp_audit "sproc_auth","all","all","on"
```

Event 80 is recorded each time an authentication check is performed:

```
sp_audit "create", "all", "testdb", "on"
go
select event, extrainfo from sybsecurity..sysaudits_01 where event = 80
go
event  extrainfo
---------------------------
80    sa_role sso_role oper_role sybase_ts_role mon_role; ; ; ;
       sso_role; ; sa/ase;
```

11.2.5.3.3.3 Auditing Stored Procedures Created with execute as owner or execute as caller

When auditing the execution of a stored procedure, if that procedure is created with `execute as owner` or `execute as caller`, the information is written to position 2 of the `extrainfo` section of the audit record.

In section 5 of the `extrainfo` column, the user name is written as either the owner or the caller, which ever applies.

Example

This is an example of an `extrainfo` column for a stored procedure that is owned by the database owner, executed by user Joe, and is created with `execute as caller`:

```
; EXECUTE AS CALLER; ; ; procedure caller = joe ; ; ;
```

Example

This is an example of an `extrainfo` column for a stored procedure that is owned by user Billy and is created with `execute as owner`:

```
; EXECUTE AS OWNER; ; ; procedure owner = billy ; ; ;
```

11.2.5.4 Granular Auditing

Granular auditing allows the auditing of options for a particular login or role. This reduces the number of audit records generated and minimizes unnecessary audits.

Granular auditing is supported for all global options and the user-specific `all` option. Records are generated for specified logins, and logins granted system roles or user defined roles. Audit records are generated only for those logins that have been granted the specified roles.

Audit Log in Attempts for Logins and Roles

In this example, there are five logins. Three are administrators and two are application logins. You can audit only the administrators who have logged in to the server by specifying the `login` option:

```
sp_audit "login", "admin1", "all", "on"
sp_audit "login", "admin2", "all", "on"
```

```
sp_audit "login", "admin3", "all", "on"
```

No audit records are generated for the application logins when they attempt to log in to the server.

You can audit at the role level for both system roles and user-defined roles, which reduces the number of audit records produced. This provides better performance, as extra processing for generating audit records for all logins is reduced. In this example, auditing is enabled to produce audit records of log-on attempts for all logins that are granted the user-defined roles `role1` and `role2`:

```
sp_audit "login", "role1", "all", "on"
sp_audit "login", "role2", "all", "on"
```

Audit all Actions

The user-specifc `all` option supports auditing for all actions of logins, and logins that are granted system roles or user-defined roles. The following audits all actions for logins that are granted the user defined role `role3`:

```
sp_audit "all", "role3", "all", "on"
```

Global Audit Options

All global audit options support auditing at login and role level (both system defined and user defined roles). For example:

```
sp_audit "config_history", "sa_role", "all", "on"
```

For a list of global options, see *Reference Manual: Procedures* > *sp_audit*.

In this example, a new login is created and granted roles `sa_role` and `oper_role`:

```
1> create login userA with password myPassWord
2> go
1> grant role sa_role to userA
2> go
1> grant role oper_role to userA
2> go
```

Auditing is enabled for the global option `dump_config` for `userA`:

```
1> sp_audit "dump_config", "userA", "all", "on"
2> go
Audit option has been changed and has taken effect immediately.
(return status = 0)
```

You can display the status of the enabled `dump_config` option using `sp_displayaudit`:

```
1> sp_displayaudit
2> go
No sprocs/triggers currently have auditing enabled.
No databases currently have default sproc/trigger auditing enabled.
No objects currently have auditing enabled.
```

```
No databases currently have default table/view auditing enabled.
Login Name     Audit Option      Value
---------- ---------------- ------------
 userA          dump_config       on
(1 row affected)
No databases currently have auditing enabled.
Audit Option     Value     Subject Name    Type of Subject
-------------  -------- ----------------- -------------------
 dump_config     on          userA              login
(1 row affected)
(return status = 0)
```

Displaying Granular Audit Options

Use `sp_displayaudit` to display the status of auditing for specific logins or roles.

To show whether the `login` option is set to enabled for login `bob`:

```
sp_displayaudit "login", "bob"

Subject Name     Audit Option     Value     Type of Subject
-------------- ---------------- --------- -----------------
bob               login        on           login
(1 row affected) (return status = 0)
```

To show whether the `login` option is enabled for the role `role1`:

```
sp_displayaudit "login", "role1"

Subject Name     Audit Option     Value     Type of Subject
-------------- ---------------- --------- -----------------
role1               login          on       role
(1 row affected)
(return status = 0)
```

Viewing Events From sybsecurity

The columns `id` and `status` are updated in `sybsecurity..sysauditoptions` to indicate audit status. For example, when the option `login` is enabled for all logins and roles, then the column `status` in `sybsecurity..sysauditoptions` is updated to 1:

```
sp_audit "login", "all", "all", "on"
select name, id, sval, status from sybsecurity..sysauditoptions where
name="login"
 name        id     sval      status
-------------------------------------
 login       NULL     on         1
```

When the audit option `login` is enabled for a system role and a user-defined role, new rows for both roles are listed in `sybsecurity..sysauditoptions` to indicate the audit status. For example, 2 is listed in the `status` column to indicate that the `id` of the stored `srid` for the specified role name:

```
sp_audit "login", "sa_role", "all", "on"
```

```
select role_id("testrole")
---------
   33
sp_audit "login", "testrole", "all", "on"
select name, id, sval, status from sybsecurity..sysauditoptions where
name="login"
 name        id     sval      status
-------------------------------------
 login      NULL    off          1
 login      3       on           0
 login      0       on           2
 login      33      on           2
```

Upgrade and Downgrade

For a minor upgrade:

- The status is indicted in the `sysauditoptions` columns `id` and `status`.
- The upgraded rows show NULL for the `id` column for all audit options except the `all` audit option.
- The `status` column uses a bit set to denote the option that need to be audited for all logins and roles for global audit options.
- For the `all` audit option, rows are added for login level and role level options are indicated with the option number 59. The `id` column in `sysauditoptions` shows the ID of the role or login, and the status bit is set to indicate role or login.

For a downgrade:

- Downgrade of granular auditing is processed by `sp_downgrade_esd`. During downgrade, the rows with status!=1, or num=59 are deleted, and `sysauditoptions` columns `id` and `status` are removed.
  The information in the `syslogins` table for the `all` audit option, and the option numbers denoting system role-level auditing, are retained. These rows are not used in version 16.0 SP03 and higher.
- The audit options `cmdtext`, `view_access`, and `table_access` maintain the login-specific audit information in `syslogins`.

For more information, see *Reference Manual* > *sp_audit* and *sp_displayaudit*.

11.2.6 Modifying the Audit System

Move the sybsecutity database to a different device, or change devices in the audit system.

11.2.6.1 Moving the sybsecurity Auditing Database

If you currently have the `sybsecurity` database on the same device as `master`, you can move `sybsecurity` to another device, with or without saving your existing global audit settings.

> **i** Note
>
> The `sybsecurity` database should be placed on its own device, separate from the `master` database. If you have more than one audit table, place each table on its own device. It can also be helpful to put each table on a separate segment which points to a separate device.

11.2.6.1.1 Moving sybsecurity Without Saving Global Audit Settings

You can move the `sybsecurity` database from the same device as `master` without saving your existing global audit settings.

Context

> **i** Note
>
> These steps include dropping the `sybsecurity` database, which destroys all audit records and global audit settings previously recorded in `sybsecurity`. Before you drop the `sybsecurity` database, make sure you archive existing records with a backup or archive the audit table to avoid losing any historical data that remain in the `sybsecurity` tables.

Procedure

1. Execute the following to stop logging audit records for all logins to the `syslogins` system table:

   ```
   sp_audit "all","all","all","off"
   ```

2. Drop the `sybsecurity` database.
3. Install `sybsecurity` again using the installation procedure described in either the configuration documentation for your platform, or *Installing Auditing with installsecurity*.
4. During the installation process, place the `sybsecurity` database on one or more devices, separate from the master device.

11.2.6.1.2 Moving sybsecurity and Saving Global Audit Settings

You can move the `sybsecurity` database from the same device as `master` and save your existing global audit settings.

Context

> **i** Note
>
> The `sybsecurity` database should be placed on its own device, separate from the `master` database. If you have more than one audit table, place each table on its own device. It can also be helpful to put each table on a separate segment which points to a separate device.

Procedure

1. Dump the `sybsecurity` database:

   ```
   dump database sybsecurity to "/remote/sec_file"
   ```

2. Drop the `sybsecurity` database:

   ```
   drop database sybsecurity
   ```

3. Initialize the first device on which you want to place the `sybsecurity` database:

   ```
   disk init name = "auditdev",
       physname = "/dev/dsk/c2d0s4",
       size = "10M"
   ```

4. Initialize the device where you want to place the `sybsecurity` log:

   ```
   disk init name = "auditlogdev",
       physname = "/dev/dsk/c2d0s5",
       size = "2M"
   ```

5. Create the new `sybsecurity` database:

   ```
   create database sybsecurity on auditdev
       log on auditlogdev
   ```

6. Load the contents of the old `sybsecurity` database into the new database. The global audit settings are preserved:

   ```
   load database sybsecurity from "/remote/sec_file"
   ```

7. Run `online database`, which upgrades `sysaudits` and `sysauditoptions` if necessary:

   ```
   online database sybsecurity
   ```

8. Load the auditing system procedures using the configuration documentation for your platform.

11.2.6.2 Deleting a Device Entry with auditinit

Delete a device entry using `auditinit`.

Procedure

1. Do one of the following:
   - (On UNIX) Source the `SYBASE.csh` or `SYBASE.sh` environment variable file.
   - (On Windows) Go to the directory `%SYBASE%\ASE-16_0\bin`
2. In a command window, enter `auditinit`.

   The following menu is displayed:

   ```
   AUDITINIT
   1. Release directory:  /usr/u/sybase
   2. Configure a Server product
   ```

3. Select `Configure a Server product`.
4. From the product list, select `Adaptive Server`, or on Windows, `SQL Server`.
5. Select `Configure a Server product`.
6. Select the server to configure.
7. Provide the sa password for the server you selected.
8. Type *Ctrl+a* and *Enter* to accept the sa account values.

   As you proceed through the menus in `auditinit`, you can change any default values that appear. As you finish each menu, type *Ctrl+a* to accept the defaults or changed values and move to the next menu.
9. From the `Sybase Server Configuration` screen, select `Configure Auditing`.

   The following menu is displayed:

   ```
   CONFIGURE AUDITING
   1.  Configure auditing:  yes
   2.  create new device for the sybsecurity database:  yes
   3.  Add a device for audit table(s)
   4.  Add a device for the audit database transaction log
   5.  Delete a device entry
   6.  Change a device entry
   ```

   Type *Ctrl+f* to scroll forward.
10. Select `Delete a Device Entry`.
11. Enter the number of the device to delete.
12. Type *Enter* key.

11.2.6.3 Changing a Device Entry with auditinit

Change a device entry using `auditinit`.

Procedure

1. Do one of the following:
   - (On UNIX) Source the `SYBASE.csh` or `SYBASE.sh` environment variable file.
   - (On Windows) Go to the directory `%SYBASE%\ASE-16_0\bin`
2. In a command window, enter `auditinit`.

   The following menu is displayed:

   ```
   AUDITINIT
   1. Release directory:  /usr/u/sybase
   2. Configure a Server product
   ```
3. Select `Configure a Server product`.
4. From the product list, select `Adaptive Server`, or on Windows, `SQL Server`.
5. Select `Configure a Server product`.
6. Select the server to configure.
7. Provide the sa password for the server you selected.
8. Type *Ctrl+a* and Enter to accept the sa account values.

   As you proceed through the menus in `auditinit`, you can change any default values that appear. As you finish each menu, type *Ctrl+a* to accept the defaults or changed values and move to the next menu.
9. From the `Sybase Server Configuration` screen, select `Configure Auditing`.

   The following menu is displayed:

   ```
   CONFIGURE AUDITING
   1.  Configure auditing:  yes
   2.  create new device for the sybsecurity database:  yes
   3.  Add a device for audit table(s)
   4.  Add a device for the audit database transaction log
   5.  Delete a device entry
   6.  Change a device entry
   ```

   Type *Ctrl+f* to scroll forward.
10. Select `Change a Device Entry`.
11. Enter the number of the device to change.

    `auditinit` displays the `Add/Change a New Device for Auditing` menu with information on the device you selected:

    ```
    ADD/CHANGE A NEW DEVICE FOR AUDITING
    1. sybsecurity physical device name:
          /secret1/sybase_dr/install/audlog
    2. Logical name of the device: aud.log
    3. size of the new device (Meg): 5
    4. Device size for auditing:5
    ```

12. Select each remaining entry that you want to change.
13. Type *Ctrl+a* to save the new entries.

11.2.7 Audit Trail Management

These general steps describe how to effectively manage the audit trail.

- Be sure that auditing is installed with two or more tables, each on a separate device. If not, consider adding additional audit tables and devices.
- Write a threshold procedure and attach it to each audit table segment.
- Set configuration parameters for the audit queue size and to indicate appropriate action should the current audit table become full.
- Query the audit trail
- Add user specified records to the audit trail

11.2.7.1 Creating more than one sysaudits table in sybsecurity

SAP strongly recommends that you not use single-device auditing for production systems. If you use only a single audit table, you create a window of time while you are archiving audit data and truncating the audit table during which incoming audit records are lost. There is no way to avoid this when using only a single audit table.

Procedure

1. Initialize the device where you want to place the additional table:

```
disk init name = "auditdev2",
    physname = "/dev/dsk/c2d0s6",
    size = "10M"
```

2. Extend the `sybsecurity` database to the device you initialized in step 1:

```
alter database sybsecurity on auditdev2 = "2M"
```

3. Run `sp_addaudittable` to create the next `sysaudits` table on the device you initialized in step 1:

```
sp_addaudittable auditdev2
```

4. Repeat steps 1 – 3 for each `sysaudits` table.

11.2.7.2 Setting Up Threshold Procedures

Before enabling auditing, establish a threshold procedure to automatically switch auditing tables when the current table is full.

> **i Note**
>
> These instructions assume that you have installed auditing with two or more tables, each on a separate device. If you have only one device for the audit tables, see *Single-Table Auditing*.

The threshold procedure for the audit device segments should:

- Make the next empty audit table current using `sp_configure` to set the `current audit table` configuration parameter.
- Archive the audit table that is almost full using the `insert...select` command.

11.2.7.2.1 Change the Current Audit Table

The `current audit table` configuration parameter establishes the table where SAP ASE writes audit rows.

As a system security officer, you can change the current audit table with `sp_configure`, using the following syntax, where `<n>` is an integer that determines the new current audit table:

```
sp_configure "current audit table", <n>
  [, "with truncate"]
```

The valid values for `<n>` are:

- 1 means `sysaudits_01`, 2 means `sysaudits_02`, and so forth.
- 0 tells the server to automatically set the current audit table to the next table. For example, if your installation has three audit tables, `sysaudits_01`, `sysaudits_02`, and `sysaudits_03`, SAP ASE sets the current audit table to:
  - 2 if the current audit table is `sysaudits_01`
  - 3 if the current audit table is `sysaudits_02`
  - 1 if the current audit table is `sysaudits_03`

The `with truncate` option specifies that the new table should be truncated if it is not already empty. If you do not specify this option and the table is not empty, `sp_configure` fails.

> **i Note**
>
> If the current audit table is truncated and you have not archived the data, the table's audit records are lost. Archive the audit data before you use the `with truncate` option.

To execute `sp_configure` to change the current audit table, you must have the `sso_role` active. You can write a threshold procedure to automatically change the current audit table.

11.2.7.2.2 Archiving the Audit Table

You can use `insert` with `select` to copy the audit data into an existing table having the same columns as the audit tables in `sybsecurity`.

Procedure

1. Create the archive database on a separate device from the one containing audit tables in `sybsecurity`.
2. Create an archive table with columns identical to those in the `sybsecurity` audit tables. If such a table does not already exist, you can use `select into` to create an empty one by having a false condition in the `where` clause. For example:

   ```
   use aud_db
   go
   select *
       into audit_data
       from sybsecurity.dbo.sysaudits_01
       where 1 = 2
   ```

   The `where` condition is always false, so an empty duplicate of `sysaudits_01` is created.

   The `select into/bulk copy` database option must be turned on in the archive database (using `sp_dboption`) before you can use `select into`.

Results

The threshold procedure, after using `sp_configure` to change the audit table, can use `insert` and `select` to copy data to the archive table in the archive database. The procedure can execute commands similar to these:

```
insert aud_db.sso_user.audit_data
select * from sybsecurity.dbo.sysaudits_01
```

11.2.7.2.3 Example Threshold Procedure for Audit Segments

This sample threshold procedure assumes that three tables are configured for auditing.

```
declare @audit_table_number int
/*
** Select the value of the current audit table
*/
select @audit_table_number = scc.value
from master.dbo.syscurconfigs scc, master.dbo.sysconfigures sc
where sc.config=scc.config and sc.name  = "current audit table"
/*
** Set the next audit table to be current.
** When the next audit table is specified as 0,
** the value is automatically set to the next one.
```

```
*/
exec sp_configure "current audit table", 0, "with truncate"
/*
** Copy the audit records from the audit table
** that became full into another table.
*/
if @audit_table_number = 1
    begin
        insert aud_db.sso_user.sysaudits
            select * from sysaudits_01
        truncate table sysaudits_01
    end
else if @audit_table_number = 2
    begin
        insert aud_db.sso_user.sysaudits
            select * from sysaudits_02
        truncate table sysaudits_02
    end
return(0)
```

11.2.7.2.4 Attach the Threshold Procedure to Each Audit Segment

Use `sp_addthreshold` to attach the threshold procedure to each audit table segment.

Before executing `sp_addthreshold`:

- Determine the number of audit tables configured for your installation and the names of their device segments
- Have the permissions and roles you need for `sp_addthreshold` for all the commands in the threshold procedure

> ⚠ Caution
>
> `sp_addthreshold` and `sp_modifythreshold` check to ensure that only a user with `sa_role` directly granted can add or modify a threshold. All system-defined roles that are active when you add or modify a threshold are inserted as valid roles for your login in the `systhresholds` table. However, only directly granted roles are activated when the threshold procedure fires.

11.2.7.2.5 Audit Tables and Their Segments

When you install auditing, `auditinit` displays the name of each audit table and its segment. The segment names are "aud_seg1" for `sysaudits_01`, "aud_seg2" for `sysaudits_02`, and so on

You can find information about the segments in the `sybsecurity` database if you execute `sp_helpsegment` with `sybsecurity` as your current database. One way to find the number of audit tables for your installation is to execute the following SQL commands:

```
use sybsecurity
go
select count(*) from sysobjects
    where name like "sysaudit%"
```

```
go
```

Get additional information about the audit tables and the `sybsecurity` database by executing the following SQL commands:

```
sp_helpdb sybsecurity
go
use sybsecurity
go
sp_help sysaudits_01
go
sp_help sysaudits_02
go
  ...
```

11.2.7.2.6 Required Roles and Permissions

To execute `sp_addthreshold`, you must be either the database owner or a system administrator.

A system security officer should be the owner of the `sybsecurity` database and, therefore, should be able to execute `sp_addthreshold`. In addition to being able to execute `sp_addthreshold`, you must have permission to execute all the commands in your threshold procedure. For example, to execute `sp_configure` for `current audit table`, the `sso_role` must be active. When the threshold procedure fires, Adaptive Server attempts to turn on all the roles and permissions that were in effect when you executed `sp_addthreshold`.

To attach the threshold procedure `audit_thresh` to three device segments:

```
use sybsecurity
go
sp_addthreshold sybsecurity, aud_seg_01, 250, audit_thresh
sp_addthreshold sybsecurity, aud_seg_02, 250, audit_thresh
sp_addthreshold sybsecurity, aud_seg_03, 250, audit_thresh
go
```

The sample threshold procedure `audit_thresh` receives control when fewer than 250 free pages remain in the current audit table.

For more information about adding threshold procedures, see *Managing Free Space with Thresholds*, in *System Administration Guide: Volume 2*.

11.2.7.2.7 Auditing with the Sample Threshold Procedure in Place

After you enable auditing, SAP ASE writes all audit data to the initial current audit table, `sysaudits_01`.

When `sysaudits_01` is within 250 pages of being full, the threshold procedure `audit_thresh` fires. The procedure switches the current audit table to `sysaudits_02`, and, immediately, SAP ASE starts writing new audit records to `sysaudits_02`. The procedure also copies all audit data from `sysaudits_01` to the `audit_data` archive table in the `audit_db` database. The rotation of the audit tables continues in this fashion without manual intervention.

11.2.7.3 Audit Configuration Parameters

Use the `audit queue size` and `suspend audit when device full` configuration parameters for your auditing installation.

Set the following configuration parameters:

- `audit queue size` sets the number of records in the audit queue in memory.
- `suspend audit when device full` determines what SAP ASE does if the current audit table becomes completely full. The full condition occurs only if the threshold procedure attached to the current table segment is not functioning properly.

11.2.7.3.1 Manage the Size of the Audit Queue

The amount of memory consumed by the audit queue pool is defined by the `audit queue size` parameter, and includes data buffers and overhead for the memory pool. However, the amount of memory in the pool can vary between releases and chip architectures.

The default audit queue size is 100 bytes.

Use `sp_configure` to set the length of the audit queue. The syntax is:

```
sp_configure "audit queue size", [<value>]
```

`value` is the number of records that the audit queue can hold. The minimum value is 1, and the maximum is 65,535. For example, to set the audit queue size to 300, execute:

```
sp_configure "audit queue size", 300
```

For more information about setting the audit queue size and other configuration parameters, see *Reference Manual: Configuration Parameters*.

11.2.7.3.2 Suspend Auditing if Devices are Full

If you have two or more audit tables, each on a separate device other than the master device, and have a threshold procedure for each audit table segment, the audit devices should never become full.

Only if a threshold procedure is not functioning properly would the “full” condition occur. Use `sp_configure` to set the `suspend audit when device full` parameter to determine what happens if the devices do become full. Choose one of these options:

- Suspend the auditing process and all user processes that cause an auditable event. Resume normal operation after a system security officer clears the current audit table.
- Truncate the next audit table and start using it. This allows normal operation to proceed without intervention from a system security officer.

Use sp_configure to set this configuration parameter. You must have the `sso_role` active. The syntax is:

```
sp_configure "suspend audit when device full",
```

```
    [0|1]
```

- `0` – truncates the next audit table and starts using it as the current audit table whenever the current audit table becomes full. If you set the parameter to 0, the audit process is never suspended; however, older audit records are lost if they have not been archived.
- `1` (the default value) – suspends the audit process and all user processes that cause an auditable event. To resume normal operation, the system security officer must log in and set up an empty table as the current audit table. During this period, the system security officer is exempt from normal auditing. If the system security officer's actions would generate audit records under normal operation, SAP ASE sends an error message and information about the event to the error log.

If you have a threshold procedure attached to the audit table segments, set `suspend audit when device full` to `1` (on). If it is set to `0` (off), SAP ASE may truncate the audit table that is full before your threshold procedure has a chance to archive your audit records.

11.2.7.4 Query the Audit Trail

To query the audit trail, use SQL to select and summarize the audit data.

If you follow the procedures discussed in *Audit Trail Management*, the audit data is automatically archived to one or more tables in another database. For example, assume that the audit data resides in a table called `audit_data` in the `audit_db` database. To select audit records for tasks performed by "bob" on July 5, 1993, execute:

```
use audit_db
go
select * from audit_data
   where loginname = "bob"
   and eventtime like "Jul 5% 93"
go
```

This command requests audit records for commands performed in the `pubs2` database by users with the system security officer role active:

```
select * from audit_data
   where extrainfo like "%sso_role%"
   and dbname = "pubs2"
go
```

This command requests audit records for all table truncations (event 64):

```
select * from audit_data
   where event = 64
go
```

To query the audit trail using the name of an audit event, use the `audit_event_name` function. For example, to request the audit records for all database creation events, enter:

```
select * from audit_data where audit_event_name(event)
   = "Create Database"
go
```

11.2.7.5 Add User-Specified Records to the Audit Trail

`sp_addauditrecord` allows users to enter comments into the audit trail. The syntax is:

```
sp_addauditrecord [<text>] [, <db_name>] [, <obj_name>]
     [, <owner_name>] [, <dbid>] [, <objid>]
```

All the parameters are optional:

- `<text>` – is the text of the message that you want to add to the `extrainfo` audit table.
- `<db_name>` – is the name of the database referred to in the record, which is inserted into the `dbname` column of the current audit table.
- `<obj_name>` – is the name of the object referred to in the record, which is inserted into the `objname` column of the current audit table.
- `<owner_name>` – is the owner of the object referred to in the record, which is inserted into the `objowner` column of the current audit table.
- `<dbid>` – is an integer value representing the database ID number of `db_name`, which is inserted into the `dbid` column of the current audit table. Do not place it in quotes.
- `<objid>` – is an integer value representing the object ID number of `obj_name`. Do not place it in quotes. `<objid>` is inserted into the `objid` column of the current audit table.

You can use `sp_addauditrecord` if:

- You have execute permission on `sp_addauditrecord`.
- The auditing configuration parameter was activated with `sp_configure`.
- The `adhoc` auditing option was enabled with `sp_audit`.

By default, only a system security officer and the database owner of `sybsecurity` can use `sp_addauditrecord`. Permission to execute it may be granted to other users.

11.2.7.5.1 Examples of Adding User-Defined Audit Records

The following examples show how to add a record to the current audit table and inserting information into the `extrainfo` and `dbname` columns.

The following example adds a record to the current audit table. The text portion is entered into the `extrainfo` column of the current audit table, “corporate” into the `dbname` column, “payroll” into the `objname` column, “dbo” into the `objowner` column, “10” into the `dbid` column, and “1004738270” into the `objid` column:

```
sp_addauditrecord "I gave A. Smith permission to view
the payroll table in the corporate database. This
permission was in effect from 3:10 to 3:30 pm on
9/22/92.", "corporate", "payroll", "dbo", 10,
1004738270
```

The following example inserts information only into the `extrainfo` and `dbname` columns of the current audit table:

```
sp_addauditrecord @text="I am disabling auditing
briefly while we reconfigure the system",
@db_name="corporate"
```

11.2.8 Truncate the Transaction Log

If you enable the `trunc log on chkpt` option for the `sybsecurity` database, you do not need to worry about the transaction log becoming full. SAP ASE truncates the log whenever it performs a checkpoint.

With this option enabled, you cannot use `dump transaction` to dump the transaction log, but you can use `dump database` to dump the database.

If you follow the procedures in *Setting Up Threshold Procedures*, audit tables are automatically archived to tables in another database. You can use standard backup and recovery procedures for this archive database.

If a crash occurs on the `sybsecurity` device, you can reload the database and resume auditing. At most, only the records in the in-memory audit queue and the current audit table are lost because the archive database contains all other audit data. After you reload the database, use `sp_configure 'current audit table', <new_current_audit_table>, 'with truncate'` to set and truncate the current audit table.

If you have not changed server-wide auditing options since you dumped the database, all auditing options stored in `sysauditoptions` are automatically restored when you reload `sybsecurity`. If not, you can run a script to set the options prior to resuming auditing.

11.2.8.1 Manage a Transaction Log with No Truncation

Attaching a last-chance threshold procedure to the transaction log segment allows you to control when the amount of space remaining on the segment is less than a threshold amount.

If you use `sp_dboption` to turn the `trunc log on chkpt` off, the transaction log may fill up. Plan to attach a *last-chance threshold procedure* to the transaction log segment. This procedure takes control when the amount of space remaining on the segment is less than a threshold amount computed automatically by SAP ASE. The threshold amount is an estimate of the number of free log pages that are required to back up the transaction log.

The default name of the last-chance threshold procedure is `sp_thresholdaction`, but you can specify a different name with `sp_modifythreshold`, as long as you have the `sa_role` active.

> **i Note**
>
> `sp_modifythreshold` checks to ensure you have "sa_role" active. See *Attaching the Threshold Procedure to Each Audit Segment* for more information.

SAP ASE does not supply a default procedure, however, *Managing Free Space with Thresholds*, in *System Administration Guide: Volume 2* contains examples of last-chance threshold procedures. The procedure should execute the `dump transaction` command, which truncates the log. When the transaction log reaches the last-chance threshold point, any transaction that is running is suspended until space is available. The suspension occurs because the option `abort xact when log is full` is always set to false for the `sybsecurity` database. You cannot change this option.

With the `trunc log on chkpt` option disabled, you can use standard backup and recovery procedures for the `sybsecurity` database, but be aware that the audit tables in the restored database may not be in sync with their status during a device failure.

11.2.9 Single-Table Auditing

If you use only a single audit table, you create a window of time while you are archiving audit data and truncating the audit table during which incoming audit records are lost. There is no way to avoid this when using only a single audit table.

SAP strongly recommends that you not use single-device auditing for production systems.

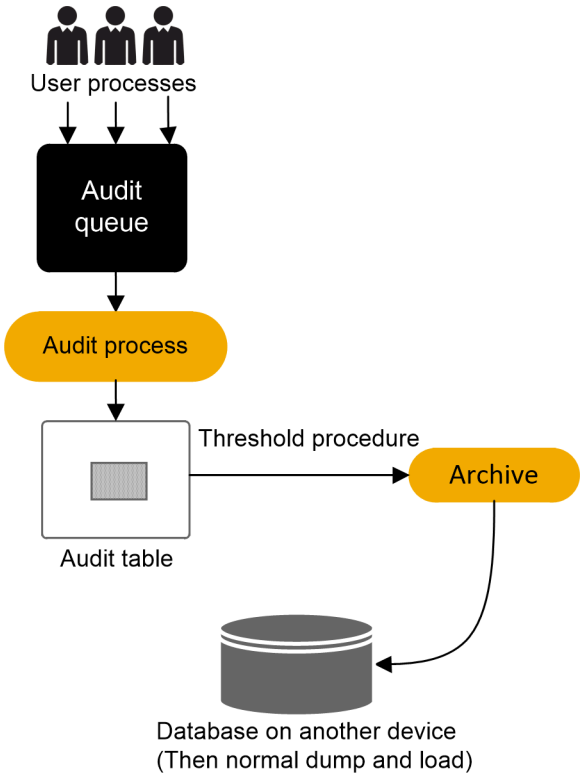
If you use only a single audit table, your audit table is likely to fill up. The consequences of this depend on how you have set `suspend audit when device full`. If you have `suspend audit when device full` set to on, the audit process is suspended, as are all user processes that cause auditable events. If `suspend audit when device full` is off, the audit table is truncated, and you lose all the audit records that were in the audit table.

For non-production systems, where the loss of a small number of audit records may be acceptable, you can use a single table for auditing, if you cannot spare the additional disk space for multiple audit tables, or you do not have additional devices to use.

The procedure for using a single audit table is similar to using multiple audit tables, with these exceptions:

- During installation, you specify only one system table to use for auditing.
- During installation, you specify only one device for the audit system table.
- The threshold procedure you create for archiving audit records is different from the one you would create if you were using multiple audit tables.

Auditing with a single audit table:



11.2.9.1 Threshold Procedure for Single-Table Auditing

The steps to configure for single-table auditing is the same as for multiple-table auditing.

For single-table auditing, the threshold procedure should:

- Archive the almost-full audit table to another table, using the `insert` and `select` commands.
- Truncate the audit table to create space for new audit records, using the `truncate table` command.

Before you can archive your audit records, create an archive table that has the same columns as your audit table. After you have done this, your threshold procedure can use `insert` with `select` to copy the audit records into the archive table.

Here is a sample threshold procedure for use with a single audit table:

```
create procedure audit_thresh as
/*
** copy the audit records from the audit table to
** the archive table
*/
insert aud_db.sso_user.audit_data
    select * from sysaudits_01
return(0)
go
/*
** truncate the audit table to make room for new
** audit records
*/
truncate table "sysaudits_01"
go
```

After you have created your threshold procedure, you will need to attach the procedure to the audit table segment. For instructions, see *Attaching the Threshold Procedure to Each Audit Segment*.

> ⚠ Caution
>
> On a multiprocessor, the audit table may fill up even if you have a threshold procedure that triggers before the audit table is full. For example, if the threshold procedure is running on a heavily loaded CPU, and a user process performing auditable events is running on a less heavily loaded CPU, the audit table may fill up before the threshold procedure triggers. The configuration parameter `suspend audit when device full` determines what happens when the audit table fills up. For information about setting this parameter, see *Suspending Auditing if Devices are Full*. .

11.2.9.2 What Happens when the Current Audit Table is Full?

These events occur when the current audit table is full.

1. The audit process attempts to insert the next audit record into the table. This fails, so the audit process terminates. An error message is written to the error log.
2. When a user attempts to perform an auditable event, the event cannot be completed because auditing cannot proceed. The user process terminates. Users who do not attempt to perform an auditable event are unaffected.
3. If you have login auditing enabled, no one can log in to the server except a system security officer.

4. If you are auditing commands executed with the `sso_role` active, the system security officer cannot execute commands.

11.2.9.3 Recover When the Current Audit Table is Full

If the current audit device and the audit queue become full, the system security officer becomes exempt from auditing.

Every auditable event performed by a system security officer after this point sends a warning message to the error log file. The message states the date and time and a warning that an audit has been missed, as well as the login name, `event` code, and other information that would normally be stored in the `extrainfo` column of the audit table.

When the current audit table is full, the system security officer can archive and truncate the audit table as described in *Archiving the Audit Table*. A system administrator can execute `shutdown` to stop the server and then restart the server to reestablish auditing.

If the audit system terminates abnormally, the system security officer can shut down the server after the current audit table has been archived and truncated. Normally, only the system administrator can execute `shutdown`.

11.3 Audit Tables

The system audit tables can be accessed only by a system security officer, who can read the tables by executing SQL commands. The only commands that are allowed on the system audit tables are `select` and `truncate`.

For content of audit tables `sysaudits_01 – sysaudits_08`, see *Reference Manual: Tables > System Tables > sysaudits_01 - sysaudits_08*.

11.3.1 Reading the extrainfo Column

The `extrainfo` column in the `sysaudits` table contains a sequence of data separated by semicolons. The data is organized in the following categories.

Table 34: Information in the extrainfo Column

Position	Category	Description
1	Roles	A list of active roles, separated by blanks.

Position	Category	Description
2	Full command text or Keywords or options	For commands supported by full text auditing, full text of an audited command. or The name of the keyword or option that was used for the event. If multiple keywords or options are listed, they are separated by commas.
3	Previous value	If the event resulted in the update of a value, this item contains the value prior to the update.
4	Current value	If the event resulted in the update of a value, this item contains the new value.
5	Other information	Additional security-relevant information that is recorded for the event. For example, parameter names and values can be recorded here.
6	Proxy information	The original login name if the event occurred while a `set proxy` was in effect.
7	Principal name	The principal name from the underlying security mechanism if the user's login is the secure default login, and the user logged in to SAP ASE via unified login. The value of this item is NULL if the secure default login is not being used.

This example shows an `extrainfo` column entry for the event of changing an auditing configuration parameter.

```
sso_role; suspend audit when device full; 1; 0; ; ; sa/ase;
```

This entry indicates that:

1 — a user with `sso_role` (system security officer)

2 — changed `suspend audit when device full`

3 — from `1` (previous value)

4 — to `0` (current value)

5 — there is no other information for this entry

6 — there is no proxy information, because `set proxy` was not in effect

7 — the principal name is sa/ase.

The other fields in the audit record give other pertinent information. For example, the record contains the server user ID (`suid`) and the login name (`loginname`).

11.3.2 Values in event and extrainfo Columns

This table includes a complete list of `sp_audit` parameters and information displayed in `event` and `extrainfo` columns of `sysaudits` tables.

> **i Note**
>
> The fourth column in this table only lists information displayed in positions 2, 3, 4, and 5 of `extrainfo` column. The information in positions 1, 6, and 7 follows the rules described in Reading the extrainfo Column [page 379].

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
(Automatically audited event not controlled by an option)	Enabling auditing with: `sp_configure "auditing", 1`	73	—
(Automatically audited event not controlled by an option)	Disabling auditing with: `sp_configure "auditing", 0`	74	—
`adhoc`	User-defined audit record created using `sp_addauditrecord`	1	Only `text` parameter value is displayed in extrainfo.
`all`	All actions of a particular user, or logins of a particular role	62	—
`allow`	`allow`	164	2 — Full text of the command
`alter`	`alter database`	2	2 — Full text of the command (for `alter role`, password is obfuscated)
	`alter index`	155	
	`alter role`	85	
	`alter table`	3	
	`alter...modify owner` (including `alter encryption key modify owner`)	132	
`bcp`	`bcp in` utility	4	—
`bind`	`sp_bindefault`	6	5 — Name of the default
	`sp_bindmsg`	7	5 — Message ID
	`sp_bindrule`	8	5 — Name of the rule
`cluster`	Cluster commands	124	—

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
`cmdtext`	Execution of a command or procedure by a specific user.	92	Only full text of command (for supported commands) or keywords are displayed in `extrainfo`. Sensitive parameters are obfuscated.

`config_history`	Configuration history using:
	`sp_audit`
	`sp_cacheconfig`
	`sp_configure`
	`sp_poolconfig`
	`sp_serveroption`
		154	5 — Includes procedure name, parameter name, old value, new value, mode (static or active), and instance ID.

`create`	`create database`	9	2 — Full text of the command (for `create role`, password is obfuscated
	`create default`	14	
	`create function`	97	
	`create index`	104	
	`create procedure`	11	
	`create role`	85	
	`create rule`	13	
	`create service`	11	
	`create table`	10	
	`create trigger`	12	
	`create view`	16	
	`sp_addmessage`	15	5 — Message number
`dbaccess`	Direct or indirect access to the database by any user	17	2 — `use cmd` (direct access using `use <dbname>`) `outside reference` (indirect access from another database)
`dbcc`	`dbcc` with all keywords	81	2 — Full text of the command
`delete`	`delete` from a table	18	2 — Full text of the command
	`delete` from a view	19	5 — Parameter names and values
`deny`	`deny`	163	2 — Full text of the command

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
`disk`	`disk init`	20	2 — Full text of the command
	`disk resize`	100	5 — Parameter names and values
	`disk mirror`	23	2 — Full text of the command
	`disk refit`	21	
	`disk reinit`	22	
	`disk remirror`	25	
	`disk unmirror`	24	
`drop`	`drop database`	26	2 — Full text of the command
	`drop default`	31	
	`drop function`	98	
	`drop index`	105	
	`drop procedure`, `drop service`	28	
	`drop role`	85	
	`drop rule`	30	
	`drop table`	27	
	`drop trigger`	29	
	`drop view`	33	
	`sp_dropmessage`	32	5 — Message number
`dump`	`dump database`	34	2 — Full text of the command (for `dump database`, password is obfuscated)
	`dump database..cumulative`	150	
	`dump transaction`	35	
`dump_config`	`dump configuration to`	149	2 — Full text of the command
`encryption_key`	`alter encryption key as | not default`	108	2 — Full text of the command (sensitive parameters are obfuscated)
	`alter encryption key add encryption`	119	
	`alter encryption key drop encryption`	120	

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
	`alter encryption key modify encryption`	118	
	`alter encryption key recover encryption`	122	
	`alter encryption regenerate key`	135	
	`create encryption key`	107	
	`drop encryption key`	109	

	`sp_encryption`	106	2 — If password is set the first time:
			`ENCR_ADMIN system_encr_passwd password ********`
			If the password is subsequently changed:
			`ENCR_ADMIN system_encr_passwd password ******** ********`

`errorlog`	`sp_errorlog`	127	5 — ERRORLOG_ADMIN, all input parameters
`errors`	Fatal error	36	5 — `<Error number>.<Severity>.<State>`
	Non-fatal error	37	
`exec_procedure`	Execution of a system stored procedure	38	5 — Values of all input parameters
`exec_trigger`	Execution of a trigger	39	—

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`

`func_dbaccess`	Access to a database using the following functions:
	`allocinfo`
	`curunreservedpgs`
	`dbencryption_status`
	`db_instanceid`
	`db_id`
	`db_name`
	`db_recovery_status`
	`dol_downgrade_check`
	`is_quiesced`
	`hadr_mode`
	`hadr_state`
	`lct_admin`
	`shrinkdb_status`
		86	2 — Name of the function

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`

`func_obj_access`	Access to an object using the following functions:
	`col_length`
	`col_name`
	`data_pages`
	`defrag_status`
	`derived_stat`
	`identity_burn_max`
	`index_col`
	`index_colorder`
	`index_name`
	`lockscheme`
	`next_identity`
	`object_id`
	`object_name`
	`object_owner_id`
	`pagesize`
	`partition_id`
	`partition_object_id`
	`partition_name`
	`reserve_identity`
	`reserved_pages`
	`row_count`
	`spaceusage`
	`used_pages`
		86	2 — Name of the function

`grant`	`grant`	40	2 — Full text of the command
	`grant role`	85	2 — Full text of the command 5 — Parameter names and values

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
`insert`	`insert` into a table	41	2 — Full text of the command
	`insert` into a view	42	5 — Parameter names and values
`install`	`installjava`	93	—
`load`	`load database`	43	2 — Full text of the command (for `load database`, sensitive parameters are obfuscated)
	`load database...cumulative`	151	
	`load transaction`	44	
`load`	`load...with [headeronly|listonly|verifyonly]`	43	—
`login`	Audits login attempts into the SAP ASE server.	45	2 — Host name and IP address of the machine from which the login was performed. For failed logins: `<Error number.Severity.State>`
`login_admin`	`alter login`	138	2 — Full text of the command (for `create login` and `alter login`, password is obfuscated)
	`create login`	103	
	`drop login`	139	
`login_locked`	Audits the hostname and network IP address when a login account is locked due to exceeding the configured number of failed login attempts	112	5 — Host machine name, host machine IP address, network password encryption information, lock reason from syslogins table
`logout`	Any logouts from the SAP ASE server	46	5 — Host name
`mount`	`mount database`	101	2 — Full text of the `mount database` command
`password`	`alter role..add passwd`	134	2 — Full text of the command (password is obfuscated)
	`alter role..drop passwd`		
	`create role with passwd`		
	`sp_passwordpolicy` and all its actions except `list`	115	5 — `PASSWORD_ADMIN` policy name and options

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`

`quiesce`	`quiesce database, prepare database`
	`quiesce database..hold security`
	`quiesce database..release`
		96
		113
		114
			2 — Full text of the command

`reference`	References between tables	91	2 — Full text of the command
`remove`	`remove java`	94	—
`revoke`	`revoke`	47	2 — Full text of the command
	`revoke role`	85	2 — Full text of the command 5 — Parameter names and values

`role`	`alter role`
	`create role`
	`drop role`
	`grant role`
	`revoke role`
		55	2 — Full text of the command (password is obfuscated for `alter role`, `create role`, and `set role`)

	`set role`	55	
`role_locked`	`alter role..lock`	133	2 — Full text of the `alter role..lock` command
`rpc`	Remote procedure call from another server	48	2 — Name of client program 5 — Server name, host name of the machine from which the RPC was executed
	Remote procedure call to another server	49	2 — Procedure name
`security`	Audit table access	61	—
	Audit table change	77	—
	Audit table truncation	58	—
	Audit option change	78	—

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
	Changes to the LDAP state	123	2 — Primary URL state and secondary URL state 3 — Previous value 4 — Current value 5 — Indicates whether the state change happened automatically or because of a manually entered command
	Security command or access to be audited, specifically, starting SAP ASE with `-u` option to unlock the administrator's account	95	5 — `Unlocking admin account`

	Server start	50	5 —
			`-d<masterdevicename>`
			`-i<interfaces file path>`
			`-S<servername>`
			`-e<errorfilename>`

	Server shutdown	51	2 — `shut down`

	`alter login`
	`create login`
	`drop login`
		138
		103
		139
			2 — Full text of the command (password is obfuscated for `alter login` and `create login`)

	`alter login profile`
	`create login profile`
	`drop login profile`
		140
		137
		141
			2 — Full text of the command

	`alter role`
	`create role`
	`drop role`
	`grant role`
	`revoke role`
		85	2 — Full text of the command (password is obfuscated for `alter role` and `create role`)

	`connect to` (CIS only)	90	2 — Full text of the `connect to` command
	`kill` (CIS only)	89	2 — Full text of the `kill` command

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
	`online database`	83	2 — Full text of the `online database` command
	`set role`	55	2 — Full text of the command (password is obfuscated)
	`set session authorization/set proxy`	88	2 — Full text of the command
	`sp_configure`	82	2 —`SETCONFIG` 5 — If a parameter is being set: number of configuration parameter If a configuration file is being used to set parameters: name of the configuration file
	`sp_hidetext`	145	5 — Total number of objects for which `sp_hidetext` was executed, and object ids from syscomments table.
	Regeneration of a password by an SSO	76	2 — `Setting SSO password` 5 — Login name
	Regeneration of asymmetric keypairs for network password encryption by the system or `sp_passwordpolicy`	117	—
	When `sp_passwordpolicy 'set', 'disallow simple passwords', 1`, and a login with a simple password is created using `create login`	125	—
	`sp_ssladmin` administration enabled	99	2 — `SSL_ADMIN addcert` if adding a certification
	`sp_webservices`	110, 111	2 — `deploy/undeploy` if deploying/undeploying a web service. `deploy_all/undeploy_all` if deploying/undeploying all web services

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`

	Built-in functions:
	`authmech`
	`show_cached_plan_in_xml`
	`show_cached_text`
	`show_cached_text_long`, `show_condensed_text`
	`show_plan`
	`valid_user`
	`workload_metric`
		86	2 — Name of the function

	`proc_role` function when executed from within a system stored procedure	80	5 — Required roles
`security_profile`	`alter login profile`	140	2 — Full text of the command
	`create login profile`	137	
	`drop login profile`	141	
`select`	`select` from a table	62	2 — Full text of the command
	`select` from a view	63	5 — Parameter names and values
`setuser`	`setuser`	84	5 — Name of the user being set
`sproc_auth`	Authorization checks that are performed inside system stored procedures. Event 80 is recorded when granular permissions are disabled. Event 146 is recorded when granular permissions are enabled.	80, 146	5 — Permission needed to run the stored procedure
`table_access`	`delete`	18	2 — Full text of the command
	`insert`	41	5 — Parameter names and values
	`select`	62	
	`update`	70	
`thread_pool`	`alter thread pool`	143	2 — Full text of the command
	`create thread pool`	142	
	`drop thread pool`	144	
`transfer_table`	`transfer table`	136	2 — Full text of the command

Audit option	Command or access to be audited	Event	Information in positions 2 — 5 of `extrainfo`
`truncate`	`truncate table`	64	2 — Full text of the command
`unbind`	`sp_unbindefault`	67	—
	`sp_unbindmsg`	69	
	`sp_unbindrule`	68	
`unmount`	`unmount database`	102	2 — Full text of the command
	`create manifest file`	116	—
`update`	`update` to a table	70	2 — Full text of the command
	`update` to a view	71	
`view_access`	`delete`	19	2 — Full text of the command
	`insert`	42	5 — Parameter names and values
	`select`	63	
	`update`	71	

11.3.3 Audit Event Values

This figure includes all audit events in ascending order.

Table 35: Audit Event Values

Audit event ID	Audited command or access	Audit event ID	Audited command or access
1	`ad hoc audit record`	81	`dbcc`
2	`alter database`	82	`sp_configure`
3	`alter table`	83	`online database`
4	`bcp in`	84	`setuser` command
5	Reserved	85	`alter role`, `create role`, `drop role`, `grant role`, or `revoke role`
6	`sp_bindefault`	86	Built-in functions
7	`sp_bindmsg`	87	Reserved

Audit event ID	Audited command or access	Audit event ID	Audited command or access
8	`sp_bindrule`	88	`set proxy` `set session authorization`
9	`create database`	89	`kill`
10	`create table`	90	`connect to`
11	`create procedure`, `create service`	91	Creation of references to tables
12	`create trigger`	92	Command text
13	`create rule`	93	`installjava`
14	`create default`	94	`remove java`
15	`sp_addmessage`	95	Unlock admin account
16	`create view`	96	`quiesce database`, `prepare database`
17	Access to database by any user	97	`create function (SQLJ)`
18	`delete` from a table	98	`drop function (SQLJ)`
19	`delete` from a view	99	`sp_ssladmin`
20	`disk init`	100	`disk resize`
21	`disk refit`	101	`mount database`
22	`disk reinit`	102	`unmount database`
23	`disk mirror`	103	`create login`
24	`disk unmirror`	104	`create index`
25	`disk remirror`	105	`drop index`
26	`drop database`	106	`sp_encryption`
27	`drop table`	107	`create encryption key`
28	`drop procedure`, `drop service`	108	`alter encryption key as | not default`
29	`drop trigger`	109	`drop encryption key`
30	`drop rule`	110	`sp_webservices` (deploy)

Audit event ID	Audited command or access	Audit event ID	Audited command or access
31	`drop default`	111	`sp_webservices` (undeploy)
32	`sp_dropmessage`	112	Login locked due to exceeding the configured number of failed attempts
33	`drop view`	113	`quiesce database..hold security`
34	`dump database`	114	`quiesce database..release`
35	`dump transaction`	115	`sp_passwordpolicy` and all of its actions except "list"
36	Fatal error	116	`create manifest file`
37	Non-fatal error	117	Regeneration of asymmetric keypairs for network password encryption by system or `sp_passwordpolicy`
38	Execution of stored procedure	118	`alter encryption key modify encryption`
39	Execution of trigger	119	`alter encryption key add encryption`
40	`grant`	120	`alter encryption key drop encryption`
41	`insert` into a table	121	Reserved
42	`insert` into a view	122	`alter encryption key recover encryption`
43	`load database`	123	LDAP state changes
44	`load transaction`	124	Cluster commands
45	`login`	125	When `sp_passwordpolicy 'set', 'disallow simple passwords', 1`, and a login with a simple password is created using `create login`.
46	`logout`	126	Reserved
47	`revoke`	127	Errorlog administration with `sp_errorlog`
48	`rpc in`	128	Reserved

Audit event ID	Audited command or access	Audit event ID	Audited command or access
49	`rpc out`	129	Reserved
50	Server start	130	Reserved
51	Server shutdown	131	Reserved
52	Reserved	132	`alter...modify owner`
53	Reserved	133	`alter role..lock`

54	Reserved	134	`alter role..add passwd`
			`alter role..drop passwd`
			`create role..with passwd`

55	Role toggling	135	`alter encryption key..regenerate key`
56	Reserved	136	`transfer table`
57	Reserved	137	`create login profile`
58	Audit table truncation	138	`alter login`
59	Reserved	139	`drop login`
60	Reserved	140	`alter login profile`
61	Access to audit table	141	`drop login profile`
62	`select` from table	142	`create thread pool`
63	`select` from view	143	`alter thread pool`
64	`truncate table`	144	`drop thread pool`
65	Reserved	145	`sp_hidetext`
66	Reserved	146	`sproc_auth`
67	`sp_unbindefault`	147	Reserved
68	`sp_unbindrule`	148	Reserved
69	`sp_unbindmsg`	149	`dump configuration to`
70	`update table`	150	`dump database..cumulative to`

Audit event ID	Audited command or access	Audit event ID	Audited command or access
71	`update view`	151	`load database..cumulative from`
72	Reserved	152	Reserved
73	Auditing enabled (`sp_configure` 'auditing', 1)	153	Reserved
74	Auditing disabled (`sp_configure` 'auditing', 0)	154	Configuration history
75	Reserved	155	`alter index`
76	Regeneration of password by an SSO	156	—
77	Audit table change	157	Reserved
78	Applies only to SAP ASE Cluster Edition	158	Reserved
79	Reserved	159	Reserved
80	Role check performed Authorization checks that are performed inside system stored procedures	163	`deny`
		164	`allow`

11.3.4 Monitoring Failed Login Attempts

When the audit option `login_locked` is enabled, the event `Locked Login (value 112)` is recorded when a login account is locked due to exceeding the configured number of failed login attempts.

To enable `login_locked`, enter:

```
sp_audit "login_locked","all","all","ON"
```

If the audit tables are full and the event cannot be logged, a message with the information is sent to the error log.

The host name and network IP address are included in the audit record. Monitoring the audit logs for the `Locked Login` event number 112 helps to identify attacks on login accounts.

11.3.5 Auditing Login Failures

Although client applications may fail to login for many reasons, SAP ASE does not provide them with any detailed information about the login failure. This is done to avoid giving information to ill intentioned users attempting to crack passwords or otherwise breach the authentication mechanisms.

However, as a system administrator, detailed information is useful for diagnosing administrative or configuration problems, and it is useful to security officers for investigating attempts to breach security.

This enables auditing for all login failures:

```
sp_audit "login", "all", "all", "fail"
```

In order to provide a barrier to inappropriate use of the information, only a user granted the SSO role can access the audit trail information containing this sensitive information.

Audits login failures for the following conditions:

- For SAP ASE started as a Windows Service, if the SQLServer service is paused (for example, by the Microsoft Management Console for Services).
- If a remote server attempts to establish a site handler for server-to-server RPCs, but insufficient resources (or any of the other conditions listed here) cause the site handler initialization to fail.
- Using SAP ASE for Windows with the Trusted Login or Unified Login configuration, but the specified user is not a trusted administrator (that is, an authentication failure).
- SAP ASE does not support the SQL interface requested by the client.
- A user is attempting to log into SAP ASE when it is in single-user mode. In single-user mode, exactly one user with the sa_role is allowed to log in to SAP ASE. Additional logins are prevented, even if they have the sa_role.
- The `syslogins` table in the master database fails to open, indicating the master database has an internal error.
- A client attempts a remote login, but `sysremotelogins` cannot be opened, or there is no entry for the specified user account and no guest account exists.
- A client attempts a remote login and, although it finds an entry referring to a local account for the specified user in `sysremotelogins`, the referenced local account does not exist.
- A client program requests a security session (for example, a Kerberos authentication), but the security session could not be established because:
  - The security subsystem was not initialized at startup.
  - Insufficient memory resources for allocated structures.
  - The authentication negotiation failed.
- An authentication mechanism is not found for the specified user.
- The specified password was not correct.
- `syslogins` does not contain the required entry for the specified login.
- The login account is locked.
- SAP ASE has reached its limit for the number of user connections.
- The configuration parameter `unified login required` is set, but the login has not been authenticated by the appropriate security subsystem.
- The network buffers are unavailable, or the requested packet size is invalid.
- A client application requests a host-based communication socket connection, but memory resources for the host-based communication buffers are not available.

- A shutdown is in progress, but the specified user does not have the sa role.
- SAP ASE could not open the default database for a login, and this login does not have access to the master database.
- A client makes a high availability login fail over request, but the high availability subsystem is does not have a high availability session for this login, or the login is unable to wait for the fail over to complete.
- A client requests a high availability login setup, but the high availability subsystem is unable to create the session or is unable to complete the TDS protocol negotiations for the high availability session.
- SAP ASE fails to setup `tempdb` for a login.
- TDS Login Protocol errors are detected.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon  : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

www.sap.com/contactsap

THE BEST RUN SAP