## Contents

- Enabling custom password checks ................................................................. 33
- Setting the login and role expiration interval for a password ................ 35
- Securing login passwords stored on disk and in memory ......................... 41
- Character set considerations for passwords ............................................. 41
- Upgrade and downgrade behavior .......................................................... 43
- Using passwords in a high-availability environment .................................. 52
- Establishing a password and login policy .................................................. 53
- Login failure .............................................................................................. 54
- Locking Adaptive Server login accounts and roles ..................................... 55
  - Locking and unlocking logins ................................................................. 55
  - Locking and unlocking login accounts .................................................. 56
  - Using syslogins to track if an account is locked .................................... 56
  - Locking and unlocking roles ................................................................. 58
  - Locking logins that own thresholds ....................................................... 59
- Managing login profiles ............................................................................. 59
  - Login profile attributes ........................................................................... 60
  - Applying login profile and password policy attributes ....................... 60
  - Creating a login profile .......................................................................... 61
  - Creating a default login profile .............................................................. 62
  - Associating a login profile with a login account .................................... 62
  - Ignoring a login profile .......................................................................... 62
  - Transfer existing login account values to a new login profile ............... 63
  - Manual replication of login profiles ....................................................... 63
  - Granting roles to login profiles .............................................................. 63
  - Invoking a login script ............................................................................ 64
  - Displaying login profile information ...................................................... 64
  - Modifying login profiles ......................................................................... 65
  - Dropping a login profile ......................................................................... 67
- Adding users to databases ........................................................................... 67
  - Adding a “guest” user to a database ....................................................... 69
  - Adding a guest user to the server ............................................................ 70
  - Adding remote users ............................................................................... 71
- Creating groups .......................................................................................... 71
  - Changing a user’s group membership ...................................................... 72
  - Setting up groups and adding users ....................................................... 72
- Using aliases in databases .......................................................................... 73
  - Adding aliases ......................................................................................... 74
  - Dropping aliases ...................................................................................... 74
  - Getting information about aliases ........................................................... 75
- Getting information about users .................................................................. 75
  - Reporting on users and processes ........................................................... 76
  - Getting information about login accounts ............................................. 77
  - Getting information about database users ............................................. 78
  - Finding user names and IDs ..................................................................... 78
Changing user information ............................................................. 79
Changing passwords ............................................................... 80
Changing user session information ......................................... 82
Dropping users and groups ............................................................ 83
Dropping users ........................................................................ 83
Dropping groups ...................................................................... 84
Monitoring license use ........................................................... 84
  How licenses are counted ....................................................... 85
  Configuring the License Use Monitor ...................................... 85
  Monitoring license use with the housekeeper task ............... 85
  Logging the number of user licenses .................................... 86
Number of user and login IDs ..................................................... 87
  Limits and ranges of ID numbers ............................................ 87
  Login connection limitations ................................................. 88
Getting information about usage: chargeback accounting ............. 89
  Reporting current usage statistics ........................................ 89
  Specifying the interval for adding accounting statistics ......... 90

CHAPTER 4  Managing Roles ............................................................... 91

Creating and assigning roles to users ...................................... 91
  System-defined roles .......................................................... 92
  System administrator privileges ............................................ 92
  System security officer privileges ........................................ 93
  Operator privileges ............................................................ 94
  Sybase Technical Support .................................................... 95
  Replication role ................................................................. 95
  Distributed Transaction Manager role ................................ 95
  High availability role ........................................................ 95
  Monitoring and diagnosis .................................................... 95
  Job Scheduler roles ........................................................... 96
  Real-time messaging role .................................................... 96
  Web Services role ............................................................. 96
  Key custodian role ............................................................. 96
  Planning user-defined roles ................................................ 96
  Creating a user-defined role ................................................. 97
  Adding and removing passwords from a role ..................... 98
Role hierarchies and mutual exclusivity .................................... 98
  Setting up default activation at login ................................ 102
  Setting conditions for role activation ................................ 103
  Dropping user-defined roles .............................................. 103
Activation and deactivating roles .............................................. 103
  Displaying information about roles .................................... 104
Granting and revoking roles ................................................... 107
  Granting roles ................................................................. 107
## Contents

Understanding grant and roles .............................................. 108  
Revoking roles ........................................................................ 109  
Roles granted to login profiles ............................................. 109  
Securing role passwords ...................................................... 109  
Character set considerations ................................................. 110  
Locked roles and syssrvroles ................................................ 110  
Login password policy checks to role passwords.................. 111  
Setting up Adaptive Server for roles ..................................... 112

### CHAPTER 5  
**External Authentication .............................................................. 117**  
Configuring Adaptive Server for network-based security .......... 118  
Security services and Adaptive Server .................................. 119  
Administering network-based security ................................... 120  
Setting up configuration files for security ......................... 121  
Identifying users and servers to the security mechanism ...... 126  
Configuring Adaptive Server for security ............................. 126  
Adding logins to support unified login .................................. 130  
Establishing Kerberos security for remote connections .......... 131  
Connecting to the server and using the security services ..... 133  
Getting information about available security services .......... 134  
Using Kerberos ...................................................................... 136  
Using principal names ........................................................... 142  
Concurrent Kerberos authentication ...................................... 147  
Configuring Adaptive Server for LDAP user authentication .... 148  
Composed DN algorithm ...................................................... 149  
Searched DN algorithm ....................................................... 149  
Configuring LDAP ............................................................... 150  
LDAP user authentication administration ............................. 151  
Adaptive Server logins and LDAP user accounts ................. 155  
Secondary lookup server support .......................................... 155  
LDAP server state transitions .............................................. 157  
LDAP user authentication tuning ......................................... 159  
Adding tighter controls on login mapping ......................... 160  
Troubleshooting LDAP user authentication errors ............ 163  
Configuring an LDAP server ............................................. 164  
LDAPS user authentication enhancements ......................... 165  
Automatic LDAP user authentication and failback ............... 166  
Setting the LDAP failback time interval ............................ 167  
Configuring Adaptive Server for authentication using PAM .... 168  
Enabling PAM in Adaptive Server ....................................... 169  
Enhanced login controls ..................................................... 172  
Forcing authentication ....................................................... 172  
Mapping logins using sp_maplogin ................................. 173
CHAPTER 6  Managing User Permissions ...................................................... 177

Overview ...................................................................................... 177
Permissions for creating databases ............................................. 179
Changing database ownership .................................................. 180
Database owner privileges ........................................................... 180
Database object owner privileges ............................................. 182
Other database user privileges .................................................... 182
Permissions on system procedures ............................................. 183
Granting and revoking permissions .............................................. 183
Object access permissions.................................................... 184
Granting permissions on dbcc commands ............................ 187
Permissions on system tables .................................................... 189
Combining grant and revoke statements .............................. 190
Understanding permission order and hierarchy .................... 192
Grant dbcc and set proxy issue warning for fipsflagscr ........ 193
Acquiring the permissions of another user ................................. 193
Using setuser ............................................................... 194
Using proxy authorization ...................................................... 195
Changing database object ownership .......................................... 199
Supported object types .......................................................... 199
Authorization ......................................................................... 200
Transferring ownership ............................................................ 200
Reporting on permissions ......................................................... 203
Querying the sysprotects table for proxy authorization ................................. 203
Displaying information about users and processes............... 204
Reporting permissions on database objects or users ........... 204
Reporting permissions on specific tables .............................. 206
Using views and stored procedures as security mechanisms..... 206
Using views as security mechanisms ......................................... 207
Using stored procedures as security mechanisms ................ 209
Understanding ownership chains .......................................... 210
Permissions on triggers .......................................................... 214
Executing a procedure with execute as owner or execute as caller 214
Creating a procedure with references to an object with an unqualified name ................................................. 218
Procedures that invoke a nested procedure in another database with a fully qualified name ........................................ 220
Using row-level access control ..................................................... 221
Access rules .............................................................................. 221
Using the Application Context Facility ................................... 230
Creating and using application contexts ................................. 233
SYS_SESSION system application context .................................. 237
Solving a problem using an access rule and ACF ............... 237
Using login triggers ................................................................. 239
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting set options from a login trigger</td>
</tr>
<tr>
<td>Setting global login triggers</td>
</tr>
<tr>
<td>CHAPTER 7</td>
</tr>
<tr>
<td>Introduction to predicated privileges</td>
</tr>
<tr>
<td>Commands used for predicated privileges</td>
</tr>
<tr>
<td>Configuring Adaptive Server to use predicated privileges</td>
</tr>
<tr>
<td>enable predicated privileges</td>
</tr>
<tr>
<td>Granting predicated privileges</td>
</tr>
<tr>
<td>Granting access to select data</td>
</tr>
<tr>
<td>Granting access to update data</td>
</tr>
<tr>
<td>Granting access to delete data</td>
</tr>
<tr>
<td>Using predicated privileges to enforce a data privacy policy</td>
</tr>
<tr>
<td>Revoking predicated privileges</td>
</tr>
<tr>
<td>How Adaptive Server saves predicated privileges in sysprotects</td>
</tr>
<tr>
<td>Predicated role activation</td>
</tr>
<tr>
<td>Combining predicates to enforce row-level privileges</td>
</tr>
<tr>
<td>Understanding SQL behavior with predicated privileges</td>
</tr>
<tr>
<td>Chain-of-ownership effect on predicated privileges</td>
</tr>
<tr>
<td>ansi_permissions and predicated privileges</td>
</tr>
<tr>
<td>Permissions on accesses made by predicates</td>
</tr>
<tr>
<td>Using triggers with predicated privileges</td>
</tr>
<tr>
<td>Recompiling predicated privileges</td>
</tr>
<tr>
<td>Disallowing recursive predicate processing</td>
</tr>
<tr>
<td>Information leakage through predicates</td>
</tr>
<tr>
<td>CHAPTER 8</td>
</tr>
<tr>
<td>Introduction to granular permissions</td>
</tr>
<tr>
<td>Configuring Adaptive Server to use granular permissions</td>
</tr>
<tr>
<td>System privileges</td>
</tr>
<tr>
<td>Effect of privileges as part of system-defined roles</td>
</tr>
<tr>
<td>Permission management</td>
</tr>
<tr>
<td>manage security permissions privilege</td>
</tr>
<tr>
<td>manage server permissions privilege</td>
</tr>
<tr>
<td>manage database permissions privilege</td>
</tr>
<tr>
<td>manage any object permission privileges</td>
</tr>
<tr>
<td>Privileges granted to system-defined roles</td>
</tr>
<tr>
<td>Privileges assigned to the database owner</td>
</tr>
<tr>
<td>Roles added with granular permissions</td>
</tr>
<tr>
<td>sa_serverprivil_role</td>
</tr>
<tr>
<td>Default roles granted to the system administrator</td>
</tr>
<tr>
<td>Limiting the power of the system administrator and database owner</td>
</tr>
</tbody>
</table>
Enable granular permissions and subsecurity ........................................... 294
Logging in to a locked-out Adaptive Server ........................................... 295
General use scenarios ........................................................................... 296
- Scenario 1: Permissions for an application server user .......... 296
- Scenario 2: Permissions for a database access manager .... 296
- Scenario 3: Permissions for a database backup manager .... 297
- Scenario 4: Permissions for a help desk operator ............... 297
- Scenario 5: Permissions for a security auditor ...................... 297
System table master.dbo.sysprotects .................................................. 298
Database user usedb_user .................................................................. 299
Grantable system privileges ................................................................. 300

CHAPTER 9 Confidentiality of Data .......................................................... 327
- Secure Sockets Layer (SSL) in Adaptive Server .................. 327
- Internet communications overview ........................................ 327
- SSL in Adaptive Server .......................................................... 330
- Enabling SSL ......................................................................... 333
- Performance .......................................................................... 340
- Cipher Suites ......................................................................... 340
- Setting SSL cipher suite preferences .................................... 341
- Using SSL to specify a common name ..................................... 347
- Specifying a common name with sp_listener ...................... 347
- Stored procedure sp_addserver changed ......................... 348
- Kerberos confidentiality ......................................................... 348
- Dumping and loading databases with password protection .... 348
- Passwords and earlier versions of Adaptive Server .............. 349
- Passwords and character sets ............................................... 349

CHAPTER 10 Auditing .............................................................................. 351
- Introduction to auditing in Adaptive Server ......................... 351
- Correlating Adaptive Server and operating system audit records 352
- The audit system ................................................................. 352
- Installing and setting up auditing ........................................... 360
- Installing the audit system .................................................... 360
- Installing auditing with auditinit ......................................... 362
- Installing auditing with installsecurity ............................... 367
- Moving the auditing database to multiple devices ............... 368
- Setting up audit trail management ....................................... 370
- Setting up transaction log management ......................... 377
- Enabling and disabling auditing ......................................... 378
- Single-table auditing ............................................................ 379
- Restarting auditing .............................................................. 382
Setting global auditing options ...................................................... 383
Auditing options: types and requirements .................................. 383
Hiding system stored procedure and command password
parameters .................................................................................. 392
Determining current auditing settings ......................................... 392
Adding user-specified records to the audit trail ......................... 393
Querying the audit trail ................................................................. 394
Understanding the audit tables ...................................................... 395
Reading the extrainfo column ...................................................... 396
Monitoring failed login attempts .................................................. 408
Auditing login failures ................................................................. 408

Index .................................................................................................. 411
CHAPTER 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.

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.

What is “information security?”

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.

Next, 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.

Adaptive Server® contains a set of security features that help you enforce your company’s security policies. For more information about security features in Adaptive Server, see Chapter 2, “Getting Started with Security Administration in Adaptive Server.”

Information security standards

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

This section describes these certifications.
Common Criteria configuration evaluation

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, Netherland, UK and the United States.

Adaptive Server version 15.0.1 completed Common Criteria validation in September, 2007. The Evaluated configuration consists of Adaptive Server version 15.0.1 with the security and directory services option. The Adaptive Server evaluation for security was carried out in accordance with the Common Criteria Evaluation and Validation Scheme (CCEVS) process and scheme. The criteria against which the Adaptive Server Enterprise 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 Adaptive Server as specified in the Supplement for Installing Adaptive Server for Common Criteria Configuration, Adaptive Server satisfies all of the security functional requirements stated in the Sybase® Adaptive Server Enterprise Security Target (Version 1.5).

Adaptive Server supports eight security functions:

- Cryptographic support – Adaptive Server supports transparent encryption of data 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 – Adaptive Server implements the discretionary access control policy over applicable database objects: databases, tables, views, stored procedures, and encryption keys.
- Identification and authentication – Adaptive Server provides its own identification and authentication mechanism 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. These functions are restricted based on discretionary access control policy rules, including role restrictions.
Information security standards

- Protection of the TOE Security Function (TSF) – Adaptive Server keeps its context separate from that of its users, and uses operating system mechanisms to ensure that memory and files used by Adaptive Server have the appropriate access settings. Adaptive Server interacts with users through well-defined interfaces designed to ensure that its security policies are enforced.

- Resource utilization – Adaptive Server provides resource limits to prevent queries and transactions from monopolizing server resources.

- Target of Evaluation (TOE) access – Adaptive Server allows authorized administrators 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.

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. SSL for Adaptive Server uses Certicom Security Builder GSE, a FIPS 140-2 level 1 validated cryptography module. See validation certificate #542, dated June 2, 2005 at the NIST Web site at http://csrc.nist.gov.

FIPS 140-2 certified Certicom Security Builder GSE is also used to encrypt login passwords in transmitted login packet, in memory and on disk, if the configuration parameter `FIPS login password encryption` is enabled.

**Note** A Security and Directory Services license is required to use SSL and to enable the FIPS login password encryption parameter. If the parameter is not enabled, OpenSSL security provider is used to perform login password encryption.

Adaptive Server encrypted columns feature relies on symmetric-key cryptography, and uses the same FIPS 140-2 validated cryptographic modules as SSL. See the *Users Guide for Encrypted Columns*.

**Note** You must have an encrypted columns license to use the Adaptive Server encrypted columns feature.
CHAPTER 2  
Getting Started with Security Administration in Adaptive Server

General process of security administration

“Performing major tasks to securely administer Adaptive Server” describes the major tasks that are required to securely administer Adaptive Server and refers you to the documentation that contains the instructions for performing each task.

❖ Performing major tasks to securely administer Adaptive Server

1. Install Adaptive Server, 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 Chapter 10, “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 Chapter 3, “Managing Adaptive Server 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 Chapter 3, “Managing Adaptive Server Logins and Database Users.”

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 Chapter 6, “Managing User Permissions.”

5 Configure encryption in your database to encrypt sensitive data in tables. Encrypt sensitive data – Configure Adaptive Server 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 Users Guide for Encrypted Columns.

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 Adaptive Server, and use the audit trail to detect penetration of the system and misuse of resources. See Chapter 10, “Auditing,” and the Adaptive Server installation and configuration documentation for your platform.

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 Chapter 5, “External Authentication” and Chapter 9, “Confidentiality of Data.”

Recommendations for setting up security

The following describes logins and how they relate to security.

- Using the “sa” login – when you install Adaptive Server, 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.

**Warning!** When logging in to Adaptive Server, 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 Adaptive Server login names that are the
  same as their respective operating system login names. This makes
  logging in to Adaptive Server easier, simplifies management of server and
  operating system login accounts, and makes it easier to correlate the audit
  data generated by Adaptive Server with that of the operating system.

### An example of setting up security

This uses special roles assigned to the users listed in Table 2-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Privilege</th>
<th>Operating system login name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rajnish Smith</td>
<td>sso_role</td>
<td>rsmith</td>
</tr>
<tr>
<td>Catherine Macar-Swan</td>
<td>sa_role</td>
<td>cmacar</td>
</tr>
<tr>
<td>Soshi Ikedo</td>
<td>sa_role</td>
<td>sikedo</td>
</tr>
<tr>
<td>Julio Rozanski</td>
<td>oper_role</td>
<td>jrozan</td>
</tr>
<tr>
<td>Alan Johnson</td>
<td>dbo</td>
<td>ajohnson</td>
</tr>
</tbody>
</table>

Table 2-2 shows the sequence of commands you might use to set up a secure
operating environment for Adaptive Server, based on the role assignments
shown in Table 2-1. After logging in to the operating system, issue these
commands using the initial “sa” account.
Security features in Adaptive Server

The major security features in Adaptive Server are:

- **“Identification and authentication”** on page 9 – ensures that only authorized users can log in to the system. In addition to password-based login authentication, Adaptive Server supports external authentication using Kerberos, LDAP, or PAM.

- **“Discretionary access control”** on page 10 – 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.

---

### Table 2-2: Examples of commands used to set up security

<table>
<thead>
<tr>
<th>Commands</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isql -Usa</code></td>
<td>Logs in to Adaptive Server as “sa.” Both <em>sa_role</em> and <em>sso_role</em> are active.</td>
</tr>
<tr>
<td><code>sp_audit &quot;security&quot;, &quot;all&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td>Sets auditing options for server-wide, security-relevant events, and the auditing of all actions that have <em>sa_role</em> or <em>sso_role</em> active.</td>
</tr>
<tr>
<td><code>sp_audit &quot;all&quot;, &quot;sa_role&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>sp_audit &quot;all&quot;, &quot;sso_role&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td><code>sp_configure &quot;auditing&quot;, 1</code></td>
<td>Enables auditing.</td>
</tr>
</tbody>
</table>

**Note** Before you enable auditing, set up a threshold procedure for the audit trail and determine how to handle the transaction log in *sybsecurity*. See Chapter 10, “Auditing.”

<table>
<thead>
<tr>
<th>Commands</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>create login</code></td>
<td>Adds logins and passwords.</td>
</tr>
<tr>
<td><code>grant role</code></td>
<td>Grant roles.</td>
</tr>
<tr>
<td><code>use sybsecurity</code></td>
<td>Grants access to the auditing database, <em>sybsecurity</em>, by making Rajnish, who is the system security officer, the database owner. Alan is not granted any system-defined roles.</td>
</tr>
<tr>
<td><code>sp_changedbowner rsmith</code></td>
<td></td>
</tr>
<tr>
<td><code>sp_locklogin sa,&quot;lock&quot;</code></td>
<td>Locks the “sa” login so that no one can log in as “sa.” Individuals can assume only the roles that are configured for them.</td>
</tr>
</tbody>
</table>

**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.
• “Division of roles” on page 11 – allows an administrator to grant privileged roles to specified users so only designated users can perform certain tasks. Adaptive Server has predefined roles, called “system roles,” such as system administrator and system security officer. In addition, Adaptive Server allows system security officers to define additional roles, called “user-defined roles.”

• “Auditing for accountability” on page 13 – 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. Adaptive Server also provides a single option to audit a set of server-wide security-relevant events.

• “Confidentiality of data” on page 13 – 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.

Identification and authentication

Adaptive Server 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 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. Adaptive Server allows both internal and external mechanisms for authentication.


External authentication

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

- 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 a user’s 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.

For more information about each of these methods of external authentication, see Chapter 5, “External Authentication.”

Managing remote servers

Internal mechanisms for administering logins and users between Adaptive Servers are described in Chapter 7, “Managing Remote Servers” in System Administration Guide, Volume I.

Discretionary access control

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. With Adaptive Server discretionary access control, you can give various permissions to users, groups, and roles using the grant command. 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.
Chapter 6, “Managing User Permissions.” Discretionary access control are discussed in

## Row-level access control

Row-level access control provides a powerful and flexible means of protecting data, down to the row level. 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 Adaptive Server 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

See “Using row-level access control” on page 221.

## 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. See “Granting Predicated Privileges” on page 251.

## Division of roles

The roles supported by Adaptive Server enable you to enforce and maintain individual accountability. Adaptive Server 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. Adaptive Server evaluates the condition expressed by the *where* clause when the role is activated, either automatically during login, or when processing the *set role* statement. See “Predicated role activation” on page 259.

**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.

See “Creating and assigning roles to users” on page 91.
Auditing for accountability

Adaptive Server 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_addaudirecord to add user-defined records to the audit trail

When you install auditing, you can specify the number of audit tables that Adaptive Server 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

Auditing functionality is discussed in Chapter 10, “Auditing.”

Confidentiality of data

Adaptive server allows you to maintain the confidentiality of data by encrypting client-server communications using the Secure Sockets Layer (SSL) standard or using 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:
Security features in Adaptive Server

- SSL – Chapter 9, “Confidentiality of Data”
- Kerberos – Chapter 5, “External Authentication”
- Encrypted columns – Encrypted Columns Users Guide
Managing Adaptive Server Logins and Database Users

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to logins and login profiles</td>
<td>16</td>
</tr>
<tr>
<td>Managing login accounts</td>
<td>16</td>
</tr>
<tr>
<td>Changing login accounts</td>
<td>19</td>
</tr>
<tr>
<td>Dropping login accounts</td>
<td>20</td>
</tr>
<tr>
<td>Choosing and creating a password</td>
<td>21</td>
</tr>
<tr>
<td>Establishing a password and login policy</td>
<td>53</td>
</tr>
<tr>
<td>Login failure</td>
<td>54</td>
</tr>
<tr>
<td>Locking Adaptive Server login accounts and roles</td>
<td>55</td>
</tr>
<tr>
<td>Managing login profiles</td>
<td>59</td>
</tr>
<tr>
<td>Adding users to databases</td>
<td>67</td>
</tr>
<tr>
<td>Creating groups</td>
<td>71</td>
</tr>
<tr>
<td>Using aliases in databases</td>
<td>73</td>
</tr>
<tr>
<td>Getting information about users</td>
<td>75</td>
</tr>
<tr>
<td>Changing user information</td>
<td>79</td>
</tr>
<tr>
<td>Dropping users and groups</td>
<td>83</td>
</tr>
<tr>
<td>Monitoring license use</td>
<td>84</td>
</tr>
<tr>
<td>Number of user and login IDs</td>
<td>87</td>
</tr>
<tr>
<td>Getting information about usage: chargeback accounting</td>
<td>89</td>
</tr>
</tbody>
</table>

*Note*  Permission requirements for operations mentioned in this chapter assume that granular permissions is disabled. Operations may differ when granular permissions is enabled. See Chapter 8, “Using Granular Permissions,” for more information on granular permissions.
Introduction to logins and login profiles

A login defines a name and a password for a user to allow access to Adaptive Server. When you execute `create login`, Adaptive Server adds a row to `master.dbo.syslogins`, assigns a unique system user ID (suid) for the new user, and fills in specified attribute information. When a user logs in, Adaptive Server 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 save time for the system security administrator because attributes of login accounts are set up and managed in one place.

Managing login accounts

The responsibility of adding new login accounts to Adaptive Server, 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.

Table 3-1 summarizes the system procedures and commands used to create and manage login accounts.

<table>
<thead>
<tr>
<th>Task</th>
<th>Required role</th>
<th>Command or procedure</th>
<th>Database, group, or role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create login accounts</td>
<td>System security officer</td>
<td><code>create login</code></td>
<td>Master database</td>
</tr>
<tr>
<td>Alter login accounts</td>
<td>System security officer</td>
<td><code>alter login</code></td>
<td>Master database</td>
</tr>
<tr>
<td>Drop login accounts</td>
<td>System security officer</td>
<td><code>drop login</code></td>
<td>Master database</td>
</tr>
<tr>
<td>Create groups</td>
<td>Database owner or system administrator</td>
<td><code>sp_addgroup</code></td>
<td>User database</td>
</tr>
<tr>
<td>Create and assign roles</td>
<td>System security officer</td>
<td><code>create role, grant role</code></td>
<td>Master database</td>
</tr>
<tr>
<td>Add users to database and assign groups</td>
<td>Database owner or system administrator</td>
<td><code>sp_adduser</code></td>
<td>User database</td>
</tr>
</tbody>
</table>
Creating login accounts

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.

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


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

The `suid` column in `syslogins` uniquely identifies each user on Adaptive 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 Adaptive Server is installed. Other users’ server user IDs are integers assigned consecutively by Adaptive Server each time `create login` is executed.

For information about choosing passwords, see Choosing and creating a password.

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.
Managing login accounts

After this statement is executed, “maryd” can log in to Adaptive 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
```

Last login and managing inactive accounts

Adaptive Server 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.

Defining a stale period

A 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:

<table>
<thead>
<tr>
<th>Value of lockreason</th>
<th>Value for locksuid</th>
<th>Description of lockreason of account</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>NULL</td>
<td>Account locked automatically due to inactivity.</td>
</tr>
</tbody>
</table>

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.
CHAPTER 3  Managing Adaptive Server Logins and Database Users

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
```

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
```

Authentication mechanisms for login

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

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

Changing 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
Dropping login accounts

- 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
```

Dropping login accounts

The command `drop login` removes an Adaptive Server 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.

Choosing and creating a password

The system security officer assigns each user a password using create login when adding the user to Adaptive Server. Users can modify their passwords at any time using the alter login statement. See “Changing passwords” on page 80.

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. Sybase recommends passwords of 8 characters or longer.

• Consist of any printable letters, numbers, or symbols.

• Be enclosed in quotation marks in create login if they:

  • Includes 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

See “Password complexity checks” on page 27.
Setting and changing the 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

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

  ```sql
  sp_passwordpolicy 'set', 'maximum failed logins', 'number'
  ```


❖ Setting the maximum failed logins for specific logins

- To set the maximum number of failed login attempts for a specific login at creation, use `create login`.

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

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


❖ Setting the maximum failed logins for specific roles

- To set the maximum failed logins for a specific role at creation, use `create role`.

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

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

  See create role in Reference Manual: Commands.
CHAPTER 3  Managing Adaptive Server Logins and Database Users

❖ Changing the maximum failed logins for specific logins
  • Use alter login to set or change the maximum failed logins for an existing login.

Changes the maximum failed logins for the login “joe” to 40:
  
  alter login joe modify max failed attempts 40

❖ Changing the maximum failed logins for specific 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

  For details on the syntax and rules for using maximum failed logins, see alter role in Reference Manual: Commands.

Logging in after losing a password
  
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 there is no way to recover a lost password.

When you start with the -p parameter, Adaptive Server generates, displays, and encrypts a random password and saves it in master.syslogins as that account’s new password.

You can use dataserver -p to reset the password for sa_role and sso_role. 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:

  dataserver -psa_role

Adaptive Server displays this message:

  New password for role 'sa_role' : qjcdyrbfkxgyc0

If sa_role does not have a password, and it is started with -psa_role, Adaptive Server prints an error message in the error log.
Choosing and creating a password

Sybase strongly recommends that you change the password for the login or role when the server restarts.

Displaying password information

This section discusses how to display password information for logins and roles.

❖ Displaying password information for specific logins

- Use `sp_displaylogin` to display the login and password settings for a login.

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
```
Displaying password information for specific roles

- Use `sp_displayroles` to display the login and password settings for a role.

This example displays information about the `physician_role` role:

```sql
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


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

This example activates the check password functionality:

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

This deactivates the check password functionality:

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

Setting and changing **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.

**Note**  Adaptive Server uses a default value of 6 for minimum password length. Sybase 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 **minimum password length** affects only new passwords created after setting the value.

❖ **Setting minimum password length for a specific login**

- To set the minimum password length for a specific login at creation, use `create login`.

  This example creates the new login “joe” with the password “Djdiek3”, and sets the **minimum password length** for “joe” to 8:

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


❖ **Setting minimum password length for a specific role**

- To set the minimum password length for a specific role at creation, use `create role`.

  This example creates the new role “intern_role” with the password “temp244” and sets minimum password length for “intern_role” 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 one of any length because **minimum password length** is set to 0.

  See `create role` in *Reference Manual: Commands*. 
Changing minimum password length for a specific login
- Use `alter login` to set or change minimum password length for an existing login.

This example changes minimum password length for the login “joe” to 8 characters.

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


Changing minimum password length for a specific role
- Use `alter role` to set or change minimum password length for an existing role.

This example sets the minimum length for “physician_role”, an existing role, to 5 characters:

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

This example 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 minimum password length for a specific login
- Use `alter login` to remove the minimum password length for an existing login.

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

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


Password complexity checks

You can use these options, which support password complexity checks, in a stored procedure interface; their values are stored in the `master.dbo.sysattributes` table.

To turn off an individual option, enter:

```
sp_passwordpolicy 'clear', option
```

To turn off all password policy options, enter:

```
sp_passwordpolicy 'clear'
```
Choosing and creating a password

Login password complexity checks are also extended to role passwords. See “Login password policy checks to role passwords” on page 111.


Disallowing simple passwords

Disallow simple password checks to see if the password contains the login name as a substring. You can set it to:

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

To set this option, enter:

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

Custom password-complexity checks

Adaptive Server allows you to 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 Adaptive Server password complexity checks, and when dropping a login, respectively. See “Enabling custom password checks” on page 33 for an example of how to create these custom stored procedures.

Specifying characters in a password

Use these 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:
CHAPTER 3 Managing Adaptive Server Logins and Database Users

- 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.
Choosing and creating a password

- 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.


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 Adaptive Server encounters a password complexity option value of -1, it interprets that as a value of 0. If an option is not set, Adaptive Server interprets the option value to be 0 as well.

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

<table>
<thead>
<tr>
<th>Password checks and policies for Adaptive Server authentication</th>
<th>Configuration parameters specified using sp_configure</th>
<th>Password complexity options specified using sp_passwordpolicy</th>
<th>Per-login overrides specified using alter login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password expiration</td>
<td>system-wide password expiration</td>
<td>system-wide password expiration</td>
<td>password expiration</td>
</tr>
<tr>
<td>Digits in password</td>
<td>check password for digit</td>
<td>min digits in password</td>
<td>N/A</td>
</tr>
<tr>
<td>Alphabetic characters in password</td>
<td>N/A</td>
<td>min alpha in password</td>
<td>N/A</td>
</tr>
<tr>
<td>Password length</td>
<td>minimum password length</td>
<td>minimum password length</td>
<td>min passwd length</td>
</tr>
<tr>
<td>Failed logins lockout</td>
<td>maximum failed logins</td>
<td>maximum failed logins</td>
<td>max failed attempts</td>
</tr>
<tr>
<td>Disallow simple passwords</td>
<td>N/A</td>
<td>disallow simple passwords</td>
<td>N/A</td>
</tr>
<tr>
<td>Special characters in password</td>
<td>N/A</td>
<td>min special char in password</td>
<td>N/A</td>
</tr>
<tr>
<td>Uppercase letters in password</td>
<td>N/A</td>
<td>min upper char in password</td>
<td>N/A</td>
</tr>
<tr>
<td>Lowercase letters in password</td>
<td>N/A</td>
<td>min lower char in password</td>
<td>N/A</td>
</tr>
<tr>
<td>Password expiration warning interval</td>
<td>N/A</td>
<td>password exp warn interval</td>
<td>N/A</td>
</tr>
<tr>
<td>Resetting your password at first login</td>
<td>N/A</td>
<td>expire login</td>
<td>N/A</td>
</tr>
<tr>
<td>Custom password complexity checks</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Set the password complexity options at the:

- Login level using create login or alter login.
- Global level using the new sp_passwordpolicy or sp_configure.

Because you can set password configuration options on a global and per-login basis, and using old and new parameters, 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
Choosing and creating a password

Examples

**Example 1**  Creates a new login and sets the minimum password length for “johnd” to 6:

```sql
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:

```sql
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”:

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

Adaptive Server checks the password 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 Adaptive Server checks the designated sequence, and the new password for login “johnd” passes these checks, the password is successfully change.

**Example 2**  If you enter the following for user “johnd”, Adaptive Server 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:

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

The check fails, and Adaptive Server prints an error message. Once one password complexity check fails, no additional options are checked.

**Example 3**  Creates a new login with the following password configuration options and sets the minimum password length for login johnd to 4:

```sql
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

Adaptive Server performs the checks 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.

Adaptive Server does not change the password and prints an error message.

To alter a password, all the checks must pass.

### Enabling custom password checks

Adaptive Server allows a system security officer to write user-defined stored procedures that enable custom password checks.

For example, 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

This user-defined stored procedure (sp_extrapwdchecks) can be called when specifying a new password to save it in an encrypted form in the pwdhistory table:

    create proc sp_extrapwdchecks
    (  
        @caller_password varchar(30),  -- the current password of caller  
    )
Choosing and creating a password

```sql
@new_password  varchar(30), -- the new password of the target acct
@loginame      varchar(30) -- user to change password on
)

as

begin

declare @current_time    datetime,
         @encrypted_pwd varbinary(30),
         @salt           varchar(8),
         @changedby     varchar(30),
         @cutoffdate    datetime

select @changedby = suser_name()
select @salt = null

-- Change this line according to your installation.
-- This keeps 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(), 'sha1'), 1, 8)
end

select @encrypted_pwd = @salt + hash(@salt + @new_password, 'sha1')

if not exists ( select 1 from master..pwdhistory
                   where name = @loginame
                   and password = @encrypted_pwd )

begin
   insert master..pwdhistory
   select @loginame, @encrypted_pwd,
       @current_time, @changedby

   return (0)
end
else
```
begin
  raiserror 22001 --user defined error message
end
end

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

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
```

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

### Setting the login and role expiration interval for a password

System administrators and system security officers can:

<table>
<thead>
<tr>
<th>Use</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>create login</td>
<td>Specify the expiration interval for a login password at creation.</td>
</tr>
<tr>
<td>alter login</td>
<td>Change the expiration interval for a login password.</td>
</tr>
<tr>
<td>create role</td>
<td>Specify the expiration interval for a role password at creation (only the system security officer can issue create role).</td>
</tr>
<tr>
<td>alter role</td>
<td>Change the expiration interval for a role password (only the system security officer can issue alter role).</td>
</tr>
</tbody>
</table>

These rules apply to password expiration for logins and roles:
Choosing and creating a password

- 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.

Password expiration turned off for pre-12.x passwords

Password expiration did not affect roles in versions earlier than Adaptive Server 12.x. In Adaptive Server 12.x and later, password expiration is deactivated for any existing user-defined role passwords.

Circumventing 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 Adaptive 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.

Creating 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.


Creating 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.
Choosing and creating a password

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`.

Changing or removing 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 5 days:

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

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


Securing login passwords on the network

Adaptive Server allows the use of asymmetric encryption to securely transmit passwords from client to server using the RSA public key encryption algorithm. Adaptive Server 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 Adaptive Server to require clients to use a login protocol. Set the Adaptive Server configuration parameter `net password encryption reqd` to require all user name- and password-based authentication requests to use RSA asymmetric encryption. See "net password encryption required," in Chapter 5, "Setting Configuration Parameters" in the System Administration Guide: Volume 1.
Adaptive Server supports two versions of the login protocol using RSA asymmetric encryption. 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.

For RSA asymmetric encryption with random number, Adaptive Server generates a new key pair:

- At each server start-up,
- Automatically at 24-hour intervals using the Adaptive Server 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"
```

**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.
Choosing and creating a password

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 Adaptive Server should do 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”


Server option “net password encryption”

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

When connecting to remote servers, Adaptive Server uses the net password encryption option to determine whether it will use password encryption.

Adaptive Server uses either RSA or Sybase proprietary algorithms 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, Adaptive Server sets net password encryption to true for sysservers entries with an ASEnterprise class value. No other server classes are modified. This improves password security between two communicating Adaptive Servers.

**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

- Sybase recommends that you use the RSA algorithm to protect passwords on the network.
- To use the RSA algorithm with random number, you must have Adaptive Server version 15.7 ESD #1 or later and use new Connectivity SDK clients version 15.7 ESD #1 or later.

Adaptive Server Enterprise
To use the RSA algorithm, without random number, you must have Adaptive Server version 15.0.2 and new Connectivity SDK clients version 15.0 ESD #7 and later.

Sybase 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 Sybase proprietary algorithm, which has been available version 12.0. Adaptive Server then uses the Sybase proprietary algorithm.

New clients that support both RSA and Sybase proprietary algorithms can set properties for both algorithms. When communicating with such clients, Adaptive Server 15.0.2 and later uses RSA encryption. A pre-15.0.2 Adaptive Server uses the Sybase proprietary algorithm.

Securing login passwords stored on disk and in memory

Login passwords used by Adaptive Server 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.

Sybase 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.

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.
Choosing and creating a password

For example, a client connects to Adaptive Server 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 Adaptive Server. Assuming the Adaptive Server 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 Adaptive Server 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.

Adaptive Server converts the clear text password 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.
Upgrade and downgrade behavior

This section contains information about upgrading and downgrading Adaptive Server between versions.

**Note** Review this section if you are upgrading to Adaptive Server version 15.7 or later from a version 15.0.

Login password downgrade

To ease the transition to the on-disk encryption algorithm when migrating from versions earlier than 15.0.2, Adaptive Server includes the password policy `allow password downgrade`. 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 Sybase proprietary algorithm used in earlier versions and the SHA-256 algorithm used in Adaptive Server 15.0.2 and later.

As long as passwords are stored in both old and new forms, you can downgrade Adaptive Server to Adaptive Server 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:

```sql
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`. 
Choosing and creating a password

- 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: login993
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 Sybase-proprietary encryption is used for this account. This login has not been used before the upgrade to Adaptive Server version 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 Adaptive Server 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 IfWpXvGLBqDgW7
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,
```
Choosing and creating a password

ver1 = 5, ver2 = 0, resetting password to QaX1RuJlCrFaE6
Old password encryption algorithm found for login name login994, suid 9,
ver1 = 5, ver2 = 0, resetting password to H1HcZdRrYcKyB2
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)

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).

Behavior changes on upgraded master database

When you upgrade the master database, Adaptive Server maintains encrypted
passwords in syslogins catalogs using algorithms from the earlier- and the
upgraded version of Adaptive Server 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.
• Adaptive Server updates the password column 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 in a new master database

In a new Adaptive Server 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'
```

### Retaining password encryption after upgrading then downgrading

If you upgrade to an Adaptive Server 15.0.2 or later, then downgrade to an earlier version, use `sp_downgrade` to retain and use the password encryption functionality from the 15.0.2 and later server. By default, Adaptive Server lets you downgrade passwords after an upgrade, until you end the password downgrade period.

**Note** Running `sp_downgrade`, shutting down the server, then restarting the same version of Adaptive Server from which you downgraded removes the changes made by `sp_downgrade`. You must re-run `sp_downgrade` to redo the changes. See the [Installation Guide](#) for information about running `sp_downgrade`.

### Adding space before you upgrade

Adaptive Server requires additional space 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 Adaptive Server 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.
Choosing and creating a password

For example, if Adaptive Server 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 Adaptive Server 15.0.2 on a new master database, or 33 additional pages if you upgraded from 15.0.1 (with allow password down grade set to 1).

The transaction log requires additional space for the updated password column. When users first log in, Adaptive Server requires about 829 2K pages per 1,000 logins, 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 Adaptive Server version 15.0.2 and later.

Downgrading

Adaptive Server supports downgrading from version 15.0.2 or later to version 15.0 or 15.0.1. If you are downgrading to an earlier version of Adaptive Server, you may need to perform additional actions.

If allow password down grade 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 “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 "prepare" to verify whether you should run sp_downgrade:

```
sp_downgrade 'prepare','15.0.1',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, Sybase 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 locks, and resets the login password for user103. The random password generated by Adaptive Server is shown only to the client who executes sp_downgrade. 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 'downgrade','15.0.1',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
Choosing and creating a password

[ ... output from other downgrade steps ..]
(return status = 0)

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

00:0000:00006:2007/05/21 05:34:07.81 server  Preparing ASE downgrade from 1502 to 1501.
00:00000:00006:2007/05/21 05:35:59.09 server  Preparing ASE downgrade from 1502 to 1501.
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, locknuid, 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 makes catalog changes, and modifies password data. The server must be in single user mode to successfully execute sp_downgrade. 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, shut down the 15.0.2 server to avoid new logins or other actions that may modify data or system catalogs. If you restart Adaptive Server at version 15.0.2 after running sp_downgrade, the earlier version shuts down and you are again upgraded to the version 15.0.2 or later level.

Expiring 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.

**Showing the current value of allow password downgrade**

To obtain the current 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
```
Choosing and creating a password

Using passwords in a high-availability environment

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

High-availability configuration

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.

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attrib Type</th>
<th>Local Value</th>
<th>Remote Value</th>
<th>Advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow password downg</td>
<td>allow password</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

(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.

**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 Adaptive Server. Login passwords continue to be used without being reset or locked by `sp_passwordpolicy` or `sp_downgrade`.

After successfully setting up high-availability environment, end the `allow password downgrade` period separately on the primary and companion servers. Similarly, downgrade to an earlier version of Adaptive Server, execute `sp_downgrade` separately on the primary and companion servers.

**Establishing a password and login policy**

Adaptive Server includes several controls for setting policies for logins, roles, and passwords for internal authentication.

In Adaptive Server, 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
Login failure

Adaptive Server must successfully authenticate a user before he or she can access data in Adaptive Server. If the authentication attempt fails, Adaptive Server returns the following message and the network connection is terminated:

```sql
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.

Adaptive Server provides the reason for the login failure 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:

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

See “Auditing login failures.”
Locking Adaptive Server login accounts and roles

To prevent a user from logging in to Adaptive Server, you can either lock or drop an Adaptive Server 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.

**Warning!** Adaptive Server 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.

Locking and unlocking logins

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

**Locking and unlocking logins**

- The system security officer can use `sp_locklogin` to manually lock or unlock a login. For example:
Locking and unlocking login accounts

Use `sp_locklogin` to lock and unlock accounts or to display a list of locked accounts. You must be a system security officer to use `sp_locklogin`.

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 on an Adaptive Server, except those with `sa_role`.
- `lock | unlock` specifies whether the account is to be locked or unlocked.

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.

Adaptive Server ensures that there is always at least one unlocked system security officer’s account and one unlocked system administrator’s account.

Using `syslogins` to track if an account is locked

`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 Adaptive Server. 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 Adaptive Server. Table 3-3 provides lockreason values and descriptions, and the value of locksuid.

**Table 3-3: The reasons and values of locksuid**

<table>
<thead>
<tr>
<th>Values for lockreason</th>
<th>Values for locksuid</th>
<th>Description of lockreason account</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>Account has not been locked.</td>
</tr>
<tr>
<td>0</td>
<td>suid of caller of sp_locklogin</td>
<td>Account locked by locksuid by manually executing sp_locklogin.</td>
</tr>
<tr>
<td>1</td>
<td>suid of caller of sp_locklogin</td>
<td>Account locked due to account inactivity, locksuid has manually executed sp_locklogin 'all', 'lock', 'ndays'.</td>
</tr>
<tr>
<td>2</td>
<td>suid of attempted login</td>
<td>Account locked by Adaptive Server due to failed login attempts reaching maximum failed logins.</td>
</tr>
</tbody>
</table>
Locking Adaptive Server login accounts and roles

Locking and unlocking 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`:

  ```sql
  alter role rolename lock
  ```

Adaptive Server includes these columns 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:

<table>
<thead>
<tr>
<th>Values for lockreason</th>
<th>Values for locksuid</th>
<th>Description of lockreason account</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>suid of caller of sp_passwordpolicy set, “allow password downgrade”, 0</td>
<td>Account locked by locksuid as the password downgrade period has ended, and login or role has not transitioned to SHA-256.</td>
</tr>
<tr>
<td>4</td>
<td>NULL</td>
<td>Account locked due to account inactivity.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values for lockreason</th>
<th>Value for locksuid</th>
<th>Description of lockreason of role</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>Role is not locked</td>
</tr>
<tr>
<td>1</td>
<td>suid of caller of alter role</td>
<td>Role locked by suid by manually executing alter role rolename lock</td>
</tr>
<tr>
<td>2</td>
<td>suid of user whose last attempted role activation led to the role getting locked</td>
<td>Role locked by Adaptive Server due to failed role activation attempts reading max failed logins.</td>
</tr>
</tbody>
</table>
**CHAPTER 3 Managing Adaptive Server Logins and Database Users**

- **Locking and unlocking roles**
  - The system security officer can use `alter role` to manually lock or unlock a role. For example:
    ```sql
    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`.
    See `alter role` in *Reference Manual: Commands*.

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

- **Locking logins that own thresholds**
  This section discusses thresholds and how they 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.

- **Managing login profiles**
  System security officers can define, alter, and drop login profiles.
  Table 3-1 summarizes the system procedures and commands used to create and manage login profiles.
Managing login profiles

### Table 3-4: Managing login profiles in Adaptive Server

<table>
<thead>
<tr>
<th>Task</th>
<th>Required role</th>
<th>Command or procedure</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create login profiles</td>
<td>System security officer</td>
<td>create login profile</td>
<td>Master database</td>
</tr>
<tr>
<td>Alter login profiles</td>
<td>System security officer</td>
<td>alter login profile</td>
<td>Master database</td>
</tr>
<tr>
<td>Drop login profiles</td>
<td>System security officer</td>
<td>drop login profile</td>
<td>Master database</td>
</tr>
<tr>
<td>Return login profile ID</td>
<td>System security officer</td>
<td>lprofile_id</td>
<td>Any database</td>
</tr>
<tr>
<td>Return login profile name</td>
<td>System security officer</td>
<td>lprofile_name</td>
<td>Any database</td>
</tr>
<tr>
<td>Display the name login profiles</td>
<td>System security officer</td>
<td>sp_displaylogin</td>
<td>Any database</td>
</tr>
<tr>
<td>Displays information about login profiles</td>
<td>System security officer</td>
<td>sp_securityprofile</td>
<td>Any database</td>
</tr>
</tbody>
</table>

### Login profile attributes

Table 3-5 summarizes the attributes of login profiles. Login profile attributes are stored in syslogins, sysloginroles and master.dbo.sysattributes.

### Table 3-5: Login Profile Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default database</td>
<td>Default database in Adaptive Server.</td>
</tr>
<tr>
<td>default language</td>
<td>Default language.</td>
</tr>
<tr>
<td>login script</td>
<td>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.</td>
</tr>
<tr>
<td>auto activated roles</td>
<td>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.</td>
</tr>
<tr>
<td>authenticate with</td>
<td>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.</td>
</tr>
<tr>
<td>track lastlogin</td>
<td>Enables last login updates.</td>
</tr>
<tr>
<td>stale period</td>
<td>Indicates the duration a login account is allowed to remain inactive before it is locked due to inactivity.</td>
</tr>
<tr>
<td>profile id</td>
<td>Specifies a database in Adaptive Server.</td>
</tr>
</tbody>
</table>

### Applying login profile and password policy attributes

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.
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

**Creating a login profile**

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

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`:

```sql
create login profile mgr_lp
```

Managing login profiles

Creating a default login profile

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


Associating a login profile 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, Adaptive Server applies password policy attributes specified by sp_passwordpolicy or default attributes. See Applying login profile and password policy attributes for information about the order in which attributes are applied.

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

create login omar_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 omar_khayyam modify login profile staff_lp

Ignoring a login profile

The ignore login profile clause is used to disable login profiles associated directly or through a default login profile. Adaptive Server follows precedence rules for applying the corresponding attributes of the login account. For more information, see Applying login profile and password policy attributes.

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

create login maryb with password itsAsecur8 login profile ignore
Transfer existing login account values to a new login profile

The following example shows 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.

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

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 Adaptive Server.

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:

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

Granting roles to login profiles

The following example creates the login profile def_lp and grants the role access_role to the login profile.

```sql
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, Adaptive Server activates the default role only if the activation predicate evaluates to true.

For information about adding or dropping auto activated roles, see “Adding or dropping auto-activated roles” on page 66.
Managing login profiles

Invoking a login script

A login script can be specified to be invoked on login through a login profile. If a global login trigger is specified through `sp_logintrigger`, the login script is invoked after the global login trigger.

```
create login profile 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.

For more information see, “Using login triggers” on page 239.

Displaying login profile information

This section discusses how to display information about login profiles.

Displaying the 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.

The following 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.

**Displaying 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.

The following 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.

**Displaying login profile binding information**

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

**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*.

**Modifying 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. See “Login Profile Attributes” on page 60 for a list of login profile attributes.
Managing login profiles

The following 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
```


Adding or dropping 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. Adaptive Server automatically activates roles 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.

Changing 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
```
Dropping a login profile

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.

The following 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

Adding 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 Adaptive Server 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 Adaptive Server 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.
Adding users to databases

If no `name_in_db` parameter is given, the name inside the database is the same as `loginame`.

**Note** This capability is different from the alias mechanism described in “Using aliases in databases” on page 73, 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. See “Changing a user’s group membership” on page 72.

`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, discussed in Chapter 6, “Managing User Permissions.”
Adding a “guest” user to a database

Creating a user named “guest” in a database enables any user with an Adaptive Server 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, Adaptive Server 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, as discussed in “Dropping users” on page 83.

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 Adaptive Server.

Note Although more than one individual can be a guest user in a database, Adaptive Server can still use the user’s server user ID, which is unique within the server, to audit each user’s activity. See Chapter 10, “Auditing.”

“guest” user permissions

“guest” inherits the privileges of “public.” 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.” See Chapter 6, “Managing User Permissions.”

When you install Adaptive Server, master..sysusers contains a guest entry.

“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
Adding users to databases

```
sp_adduser guest
revoke all on titles from guest
```

“guest” user in installed system databases
Adaptive Server 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.

“guest” user in pubs2 and pubs3
The “guest” user entry in the sample databases allows new Adaptive Server 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

Adding a guest user 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.

**Warning!** 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. Sybase 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.
Adding remote users

You can allow users on another Adaptive 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 Chapter 7, “Managing Remote Servers” in System Administration Guide, Volume I.

Creating 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. They are especially useful if you administer an Adaptive Server 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.
Creating groups

Changing a user’s 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.

Setting up groups and adding 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. See “Changing a user’s group membership” on page 72.
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”).

**Using 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 Adaptive 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 Adaptive Server account, to the database user name “vp.”

**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. See Chapter 10, “Auditing.”

The collective user identity from using aliases implies set-ownership for database objects. For example, if user “loginA” is aliased to dbo in in database db1, all objects created by “loginA” in db1 are owned by dbo. However, Adaptive Server concretely records an object’s ownership in terms of the login name and the creator’s database user ID. See “Concrete identification” on page 185. An alias cannot be dropped from a database if he or she concretely owns objects in that database.

**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.
Adding aliases

To add an alias for a user, use `sp_addalias`:

```
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 in Adaptive Server 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.

Executing `sp_addalias` maps the user name specified by `loginame` to the user name specified by `name_in_db`. It does this by adding a row to the system table `sysalternates`.

When a user tries to use a database, Adaptive Server checks for the user’s server user ID number (`suid`) in `sysusers`. If it is not found, Adaptive Server then checks `sysalternates`. 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 on Adaptive Server but are not authorized to use Mary’s database. Mary executes the following commands:

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

**Warning!** 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.

Dropping aliases

Use `sp_dropalias` to drop the mapping of an alternate `suid` to a user ID. Doing this 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.

Getting information about aliases

To display information about aliases, use sp_helpuser. For example, to find the aliases for “dbo,” execute:

```
sp_helpuser dbo
```

<table>
<thead>
<tr>
<th>Users_name</th>
<th>ID_in_db</th>
<th>Group_name</th>
<th>Login_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>1</td>
<td>public</td>
<td>sa</td>
</tr>
</tbody>
</table>

(1 row affected)

Users aliased to user.
Login_name

----------------------
andy
christa
howard
linda

Getting information about users

Table 3-6 lists procedures you can use to obtain information about users, groups, and current Adaptive Server usage.
Getting information about users

Table 3-6: Reporting information about Adaptive Server users and groups

<table>
<thead>
<tr>
<th>Task</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report current Adaptive Server users and processes</td>
<td>sp_who</td>
</tr>
<tr>
<td>Display information about login accounts</td>
<td>sp_displaylogin</td>
</tr>
<tr>
<td>Report users and aliases in a database</td>
<td>sp_helpuser</td>
</tr>
<tr>
<td>Report groups within a database</td>
<td>sp_helpgroup</td>
</tr>
</tbody>
</table>

Reporting on users and processes

Use sp_who to report information about current users and processes on Adaptive Server:

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

where:

- `loginame` – is the user’s Adaptive Server 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:

```
<table>
<thead>
<tr>
<th>fid</th>
<th>spid</th>
<th>status</th>
<th>loginame</th>
<th>origname</th>
<th>hostname</th>
<th>blk_spid</th>
<th>dbname</th>
<th>tempdbname</th>
<th>cmd</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>running</td>
<td>sa</td>
<td>sa</td>
<td>sunbird</td>
<td>0</td>
<td>pubs2</td>
<td>tempdb</td>
<td>SELECT</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>master</td>
<td>0</td>
<td></td>
<td>tempdb</td>
<td>NETWORK HANDLER</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>master</td>
<td>0</td>
<td></td>
<td>tempdb</td>
<td>MIRROR HANDLER</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>master</td>
<td>0</td>
<td></td>
<td>tempdb</td>
<td>AUDIT PROCESS</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>master</td>
<td>0</td>
<td></td>
<td>tempdb</td>
<td></td>
</tr>
</tbody>
</table>
```
Getting 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:

```
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
```
Getting information about users

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)

Getting 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`:

<table>
<thead>
<tr>
<th>Users_name</th>
<th>ID_in_db</th>
<th>Group_name</th>
<th>Login_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>1</td>
<td>public</td>
<td>sa</td>
</tr>
<tr>
<td>marcy</td>
<td>4</td>
<td>public</td>
<td>marcy</td>
</tr>
<tr>
<td>sandy</td>
<td>3</td>
<td>public</td>
<td>sandy</td>
</tr>
<tr>
<td>judy</td>
<td>5</td>
<td>public</td>
<td>judy</td>
</tr>
<tr>
<td>linda</td>
<td>6</td>
<td>public</td>
<td>linda</td>
</tr>
<tr>
<td>anne</td>
<td>2</td>
<td>public</td>
<td>anne</td>
</tr>
<tr>
<td>jim</td>
<td>7</td>
<td>senioreng</td>
<td>jim</td>
</tr>
</tbody>
</table>

Finding user names and IDs

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

<table>
<thead>
<tr>
<th>To find</th>
<th>Use</th>
<th>With the argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server user ID</td>
<td>suser_id</td>
<td>(&quot;server_user_name&quot;)</td>
</tr>
<tr>
<td>Server user name (login name)</td>
<td>suser_name</td>
<td>(server_user_ID)</td>
</tr>
</tbody>
</table>

The arguments for these system functions are optional. If you do not provide one, Adaptive Server displays information about the current user.
This example shows how to find the server user ID for the user “sandy:”

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

---

3

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

```sql
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 3-8: System functions user_id and user_name**

<table>
<thead>
<tr>
<th>To find</th>
<th>Use</th>
<th>With the argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td><code>user_id</code></td>
<td>(“db_user_name”)</td>
</tr>
<tr>
<td>User name</td>
<td><code>user_name</code></td>
<td>(“db_user_ID”)</td>
</tr>
</tbody>
</table>

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

```sql
select user_name(10)
```

```sql
NULL
(1 row affected)
```

```sql
select user_name( )
```

```sql
dbo
(1 row affected)
```

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

```sql
NULL
(1 row affected)
```

---

**Changing user information**

Table 3-9 lists the system procedures you use to change passwords, default database, default language, full name, or group assignment.
Changing user information

Table 3-9: Commands or system procedures for changing user information

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

Changing 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 itsAsecret


Requiring new passwords

You may choose to use the systemwide password expiration configuration parameter to establish a password expiration interval, which forces all Adaptive Server users to change their passwords on a regular basis. See Chapter 5, "Setting Configuration Parameters,” in the System Administration Guide: Volume 1. Even if you do not use systemwide password expiration, it is important, for security reasons, that users change their passwords periodically.

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, 2007 at 10:30 a.m., with a password expiration interval of 30 days, the password expires on August 31, 2007 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, 2007:

```sql
select name, pwdate
from syslogins
where pwdate < "Sep 15 2007"
```

Null passwords

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

```sql
alter login sa with password null modify password 8M4LNc
```

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

Logging in after lost password

You can use `dataserver -plogin_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, Adaptive Server 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:

```bash
dataserver -prsmith
```
Changing user information

To reset the password of the sso_role:

```
  dataserver -psso_role
```

Changing 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 Adaptive 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 Adaptive Server.

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

For example, if a user logs in to Adaptive Server 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:

```sql
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:

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

### Dropping 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 an Adaptive Server 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`:

```sql
sp_dropuser name_in_db
```

You cannot drop a user who owns objects. Since there is no command to 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.
Monitoring license use

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. To remove users from a group, use sp_change_group, discussed in “Changing a user’s group membership” on page 72.

Monitoring license use

The License Use Monitor allows a system administrator to monitor the number of user licenses used in Adaptive Server, and to securely manage the license agreement data. That is, you can ensure that the number of licenses used on your Adaptive 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 Sybase sales representative.

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

See “Configuring the License Use Monitor,” below.
How licenses are counted

A license is the combination of a host computer name and a user name. If a user logs in to Adaptive Server 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 to Adaptive Server from the same host, but with different user names, each distinct combination of user name and host name uses one license.

Configuring 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.

Monitoring 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 Adaptive Server. 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 Sybase Sales for additional licenses.
Monitoring license use

The housekeeper chores task runs during Adaptive Server 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 Chapter 3, “Using Engines and CPUs,” in the Performance and Tuning Series:Basics.

Logging the number of user licenses

The syblicenseslog system table is created in the master database when you install or upgrade Adaptive Server. The License Use Monitor updates the columns in syblicenseslog at the end of each 24-hour period, as shown in Table 3-10.

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>-1 – housekeeper cannot monitor licenses.</td>
</tr>
<tr>
<td></td>
<td>0 – number of licenses not exceeded.</td>
</tr>
<tr>
<td></td>
<td>1 – number of licensees exceeded.</td>
</tr>
<tr>
<td>logtime</td>
<td>Date and time the log information was inserted.</td>
</tr>
<tr>
<td>maxlicenses</td>
<td>Maximum number of licenses used during the previous 24 hours.</td>
</tr>
</tbody>
</table>

syblicenseslog looks similar to this:

<table>
<thead>
<tr>
<th>status</th>
<th>logdate</th>
<th>maxlicenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Jul 17 1998 11:43AM</td>
<td>123</td>
</tr>
<tr>
<td>0</td>
<td>Jul 18 1998 11:47AM</td>
<td>147</td>
</tr>
<tr>
<td>1</td>
<td>Jul 19 1998 11:51AM</td>
<td>154</td>
</tr>
<tr>
<td>0</td>
<td>Jul 20 1998 11:55AM</td>
<td>142</td>
</tr>
<tr>
<td>0</td>
<td>Jul 21 1998 11:58AM</td>
<td>138</td>
</tr>
<tr>
<td>0</td>
<td>Jul 21 1998 3:14PM</td>
<td>133</td>
</tr>
</tbody>
</table>

In this example, the number of user licenses used exceeded the limit on July 19, 1998.

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

The second row for July 21, 1998 was caused by a shutdown and restart of the server.
Number of user and login IDs

Adaptive Server supports over 2,000,000,000 logins per server and users per database. Adaptive Server uses negative numbers as well as positive numbers to increase the range of possible numbers available for IDs.

Limits and ranges of ID numbers

Table 3-11 describes the valid ranges for the ID types.

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

Figure 3-1 illustrates the limits and ranges for logins, users, and groups.

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).
Login connection limitations

Although Adaptive Server allows you to define more than two billion logins per server, the actual number of users that can connect to Adaptive Server at one time is limited by the:

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

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

- **Allowing the maximum number of logins and simultaneous connections**
  1. Configure the operating system on which Adaptive Server is running for at least 32,000 file descriptors.
  2. Set the value of number of user connections to at least 32,000.

  **Note** Before Adaptive Server can 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.

### Table 3-12: Global variables for logins, users, and groups

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

To issue a global variable, enter:

```
select variable_name
```

For example:
Getting information about usage: chargeback accounting

When a user logs in to Adaptive Server, the server begins accumulating CPU and I/O usage for that user. Adaptive Server can report total usage 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 on Adaptive Server.

Displaying current accounting totals

sp_reportstats displays current accounting totals for Adaptive Server 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

Adaptive Server accumulates CPU and I/O statistics until you clear the totals from syslogins by running sp_clearstats. sp_clearstats initiates a new accounting interval for Adaptive Server 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 Adaptive Server CPU and I/O usage, run sp_clearstats once a month.

For detailed information about these stored procedures, see the Reference Manual: Procedures.
Getting information about usage: chargeback accounting

Specifying 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 in Adaptive Server:

```
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 an Adaptive Server session.

The minimum value allowed for either configuration parameter is 1. The maximum value allowed is 2,147,483,647.
CHAPTER 4

Managing Roles

This chapter includes information about using roles in Adaptive Server.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating and assigning roles to users</td>
<td>91</td>
</tr>
<tr>
<td>Granting and revoking roles</td>
<td>107</td>
</tr>
<tr>
<td>Securing role passwords</td>
<td>109</td>
</tr>
</tbody>
</table>

**Note**  Permission requirements for operations mentioned in this chapter assume that granular permissions is disabled. Operations may differ when granular permissions is enabled. See Chapter 8, “Using Granular Permissions,” for more information on granular permissions.

Creating and assigning roles to users

Roles are collections of privileges that allow the role assignee to perform their job. The roles supported by Adaptive Server let you enforce individual accountability. Adaptive Server 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. For more information on permissions, see Chapter 6, “Managing User Permissions.”
Creating and assigning roles to users

System-defined roles

Table 4-1 lists the system roles, the value to use for the role\_granted option of the grant role or revoke role command, and the tasks usually performed by a person with that role.

Note Each role is described in detail in the following sections.

<table>
<thead>
<tr>
<th>Role</th>
<th>Value for role_granted</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System administrator</td>
<td>sa_role</td>
<td>Manage and maintain Adaptive Server databases and disk storage</td>
</tr>
<tr>
<td>System security officer</td>
<td>sso_role</td>
<td>Perform security-related tasks</td>
</tr>
<tr>
<td>Operator</td>
<td>oper_role</td>
<td>Back up and load databases server-wide</td>
</tr>
<tr>
<td>Sybase Technical Support</td>
<td>sybase_ts_role</td>
<td>Analysis and repair of database structures</td>
</tr>
<tr>
<td>Replication</td>
<td>replication_role</td>
<td>Replicate user data</td>
</tr>
<tr>
<td>Distributed transaction manager</td>
<td>dtm_tm_role</td>
<td>Coordinate transactions across servers</td>
</tr>
<tr>
<td>High availability</td>
<td>ha_role</td>
<td>Administer and execute failover</td>
</tr>
<tr>
<td>Monitor and diagnosis</td>
<td>mon_role</td>
<td>Administer and execute performance and diagnostic monitoring</td>
</tr>
<tr>
<td>Job Scheduler administration</td>
<td>js_admin_role</td>
<td>Administer Job Scheduler</td>
</tr>
<tr>
<td>Job Scheduler user</td>
<td>js_user_role, js_client_role</td>
<td>Create and run jobs through Job Scheduler</td>
</tr>
<tr>
<td>Real-time messaging</td>
<td>messaging_role</td>
<td>Administer and execute real-time messaging</td>
</tr>
<tr>
<td>Web Services</td>
<td>webservices_role</td>
<td>Administer Web services</td>
</tr>
<tr>
<td>Key custodian</td>
<td>keycustodian_role</td>
<td>Create and manage encryption keys</td>
</tr>
</tbody>
</table>

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.

System administrator privileges

System administrators:

- Handle tasks that are not application-specific
- Work outside the Adaptive Server discretionary access control system
The role of system administrator is usually granted to individual Adaptive Server 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 Adaptive Server. 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 in Adaptive Server, 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
Creating and assigning roles to users

- 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 the *Users Guide for Encrypted Columns*). 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. See “Creating and assigning roles to users” on page 91.

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`
Sybase Technical Support

A Sybase 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. Sybase recommends that the system security officer grant this role to a knowledgeable Sybase 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 Adaptive Server 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.
Creating and assigning roles to users

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 the Encrypted Columns Users Guide.

Planning user-defined 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. Adaptive Server 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.

Creating a user-defined role

A user with sso_role uses the create role command to create a role. See create role in the the Reference Manual: Commands.

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.
Creating and assigning roles to users

Adding and removing 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
```

**Note** When you assign a password to a role, any user granted the role must specify the password to Adaptive Server at the time of activating the role.

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.

Defining and changing mutual exclusivity of roles

To define mutual exclusivity between two roles, use:

```
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
```

Defining and changing 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
```
Creating and assigning roles to users

```
grant role oper_role to super_user
```

**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 Figure 4-1, 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 Figure 4-1, “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.

*Figure 4-1: Explicitly and implicitly granted privileges*

- You cannot grant a role to another role that is contained by the first role. This prevents “loops” within the hierarchy.

  For example, in Figure 4-2, you cannot grant the “specialist” role to the “consultant” role; “consultant” is already contained in “specialist.”
Figure 4-2: Granting a role to a role contained by grantor

- 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 Figure 4-3, 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.”

Figure 4-3: Mutual exclusivity at membership

- The user can activate or deactivate only directly granted roles.

For example, in the hierarchy shown in Figure 4-3, 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. See “Activating and deactivating roles” on page 103.
Creating and assigning roles to users

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.”

Setting up 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 Adaptive Server, 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
```
CHAPTER 4    Managing Roles

Setting 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. Adaptive Server 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. See “Predicated role activation” on page 259.

Dropping user-defined roles

As system security officer, drop a role using:

```
drop role role_name [with override]
```

where `role_name` is the name of a user-defined role.

The `with override` option revokes all access privileges granted to the role in every database on the server.

If you do not use 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.

Activating and deactivating roles

Roles must be active to have access privileges (that is, inactive roles do not have 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
```
Creating and assigning roles to users

If the role was granted using an activation predicate, Adaptive Server evaluates the predicate at this time. 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.

Displaying information about roles

Table 4-2 lists the system procedures and functions to use to find information about roles.

<table>
<thead>
<tr>
<th>To display information about</th>
<th>Use</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>The role ID of a role name</td>
<td>role_id system function</td>
<td>“Finding role IDs and names” on page 104</td>
</tr>
<tr>
<td>The role name of a role ID</td>
<td>role_name system function</td>
<td>“Finding role IDs and names” on page 104</td>
</tr>
<tr>
<td>System roles</td>
<td>show_role system function</td>
<td>“Viewing active system roles” on page 105</td>
</tr>
<tr>
<td>Role hierarchies and roles that have been granted to a user or users</td>
<td>sp_displayroles system procedure</td>
<td>“Displaying a role hierarchy” on page 105</td>
</tr>
<tr>
<td>Whether one role contains another role in a role hierarchy</td>
<td>role_contain system function</td>
<td>“Viewing user roles in a hierarchy” on page 105</td>
</tr>
<tr>
<td>Whether two roles are mutually exclusive</td>
<td>mut_excl_roles system function</td>
<td>“Determining mutual exclusivity” on page 106</td>
</tr>
<tr>
<td>Roles that are active for the current session</td>
<td>sp_activeroles system procedure</td>
<td>“Determining role activation” on page 106</td>
</tr>
<tr>
<td>Whether you have activated the correct role to execute a procedure</td>
<td>has_role system function</td>
<td>“Checking for roles in stored procedures” on page 106</td>
</tr>
<tr>
<td>Logins, including roles that have been granted</td>
<td>sp_displaylogin system procedure</td>
<td>“Getting information about login accounts” on page 77</td>
</tr>
<tr>
<td>Permissions for a user, group, or role</td>
<td>sp_helprotect system procedure</td>
<td>“Reporting on permissions” on page 203</td>
</tr>
</tbody>
</table>

Finding role IDs and names

To find a role ID when you know the role name, use:

\[
\text{role_id}(\text{role_name})
\]
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`.

### Viewing active system roles

Use `show_role` to display the currently active system roles for the specified login:

```
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`.

---

**Note** The `show_role` function does not include information about user-defined roles.

### Displaying a 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:

```
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.

### Viewing user roles in a hierarchy

Use `role_contain` to determine whether any role you specify contains any other role you specify:

```
role_contain ("role1", "role2")
```

If `role1` is contained by `role2`, `role_contain` returns 1.
Creating and assigning roles to users

Any user can execute role_contain.

Determining 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:

```
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.

Determining role activation

To find all active roles for the current login session of Adaptive Server, use:

```
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.

Checking for roles in stored procedures

Use has_role within a stored procedure to guarantee that only users with a specific role can execute the procedure. Only has_role provides a fail-safe way to prevent 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 can never be granted permission to execute a stored procedure, use has_role within the stored procedure itself 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 1 if the invoker possesses it. Otherwise, it returns 0.

For example, here is a procedure that uses has_role to see if the user has the sa_role 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 System Administrator role"
    return 0

Granting and revoking roles

After a role is defined, it can be granted to any login account or role in the server, provided that it does not violate the rules of mutual exclusivity and hierarchy. Table 4-3 lists the tasks related to roles, the role required to perform the task, and the command to use.

<table>
<thead>
<tr>
<th>Task</th>
<th>Required role</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grant the sa_role role</td>
<td>System administrator</td>
<td>grant role</td>
</tr>
<tr>
<td>Grant the sso_role role</td>
<td>System security officer</td>
<td>grant role</td>
</tr>
<tr>
<td>Grant the oper_role role</td>
<td>System security officer</td>
<td>grant role</td>
</tr>
<tr>
<td>Grant user-defined roles</td>
<td>System security officer</td>
<td>grant role</td>
</tr>
<tr>
<td>Create role hierarchies</td>
<td>System security officer</td>
<td>grant role</td>
</tr>
<tr>
<td>Modify role hierarchies</td>
<td>System security officer</td>
<td>revoke role</td>
</tr>
<tr>
<td>Revoke system roles</td>
<td>System security officer</td>
<td>revoke role</td>
</tr>
<tr>
<td>Revoke user-defined roles</td>
<td>System security officer</td>
<td>revoke role</td>
</tr>
</tbody>
</table>

Granting roles

To grant roles to users, roles, or login profiles, use:

grant role role_name
    [where pred_expression]
    to {username | rolename | login_profile_name}

where:
Granting and revoking roles

- role_name – is the name of a system- or user-defined role that the system security officer is granting to a user or a role.

- where pred_expression – is a SQL condition that must be satisfied when the named role is activated. pred_expression may be used only when granting a role to username or login_profile_name.

- to username | rolename | login_profile_name – identifies the name of the login, role or login profile to which you are granting the role.

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

For more information, see “Predicated role activation” on page 259.

Understanding grant 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. See “Displaying information about roles” on page 104.

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.
CHAPTER 4  Managing Roles

Revoking roles

Use `revoke role` to revoke roles from users, other roles, and login profiles:

```sql
revoke role role_name [, role_name]...from grantee [, grantee]...
```

where:

- `role_name` – is the role being revoked. You can specify any number of roles to be revoked.
- `grantee` – is the name of the user or role. You can specify any number of grantees.

All roles listed in the `revoke` statement are revoked from all grantees.

Roles granted to login profiles

A role granted to a login profile can be activated by any user assigned that profile. See “Granting roles to login profiles” on page 63.

Securing role passwords

In versions of Adaptive Server earlier than 15.7, role passwords were stored using Sybase-proprietary encryption in the `syssrvroles` system table. As of Adaptive Server version 15.7, role passwords are stored securely on-disk as SHA-256 digests.

When you upgrade Adaptive Server to version 15.7 or later, and activate a role password for the first time after the upgrade, Adaptive Server encrypts the role password and stores it as an SHA-256 digest.

You cannot downgrade a role password that has been encrypted in SHA-256; instead, upon downgrade, Adaptive Server truncates the role password and locks the role. The administrator must then reset the password and unlock the role after the downgrade.

**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.
Securing role passwords

Character set considerations

In versions of Adaptive Server earlier than 15.7, passwords used the server’s default character set before they were encrypted. This changed, and in Adaptive Server now automatically converts passwords 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.

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. Adaptive Server stores information about locked roles 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.

Adaptive Server resets these to NULL when a role is unlocked.

The values and descriptions are:

<table>
<thead>
<tr>
<th>Values for lockreason</th>
<th>Value for locksuid</th>
<th>Description of lockreason of role</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>Role has not been locked</td>
</tr>
<tr>
<td>1</td>
<td>suid of caller of alter role</td>
<td>Role locked by suid by manually executing alter role rolename lock</td>
</tr>
<tr>
<td>2</td>
<td>suid of user whose last attempted role activation led to the role getting locked</td>
<td>Role locked by Adaptive Server due to failed-role activation attempts reaching maximum number of failed logins</td>
</tr>
</tbody>
</table>

Note: If you are using high availability functionality, both the primary and companion servers are updated when you update the syssrvroles columns.
Login password policy checks to role passwords

In Adaptive Server version 15.7, the 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

High-availability support for password policy options

The Adaptive Server high-availability functionality synchronizes these password policy options between primary and secondary servers:

- 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
Securing role passwords

- keypair regeneration period
- keypair error retry wait
- keypair error retry count

Adaptive Server 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:

```sql
sp_companion "MONEY1", do_advisory, 'all'
go
```

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Attrib Type</th>
<th>Local Value</th>
<th>Remote Value</th>
<th>Advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>expire login</td>
<td>password po</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>maximum failed</td>
<td>password po</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>min alpha in pa</td>
<td>password po</td>
<td>10</td>
<td>12</td>
<td>2</td>
</tr>
</tbody>
</table>

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.

Setting up Adaptive Server for roles

Installing

Before using the role functionality, make sure Adaptive Server 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
CHAPTER 4    Managing Roles

Upgrading Adaptive Server

During the upgrade process, Adaptive Server automatically adds 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.

Downgrading Adaptive Server

When you downgrade Adaptive Server to version 15.5, Adaptive Server truncates and locks role passwords. In addition, Adaptive Server 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, Adaptive Server:

- 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. 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 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 'downgrade','15.5',1
2> go
Downgrade from 15.7.0.0 to 15.5.0.0 (command: 'downgrade')

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 comman 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
Securing role passwords

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 lockuid = null, lockreason = null, lockdate = null where lockuid is not null or lockreason is not null or lockdate is not null delete syscolumns where id = object_id('syssrvroles') and name in ('lockuid', '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)

Additional messages appear in the error log to identify steps that occurred during sp_downgrade and any system errors that may occur, such as in this example of error log output for the example downgrade procedure:

00:00:06:00000:00006:2011/06/28 06:21:23.95 server Preparing ASE downgrade from 15.7.0.0
After running `sp_downgrade`, shut down the server to avoid new logins or other actions that may modify data or system catalogs.

If you restart Adaptive Server at version 15.7:

- After successfully executing `sp_downgrade` and shutting down the server, Adaptive Server performs internal upgrade actions again, and any changes to system tables are upgraded to version 15.7.

- Before starting an earlier version of Adaptive Server to which you are reverting, you must execute `sp_downgrade` again.

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:

```sql
select srid, status, name, password from syssrvroles
```

```sql
go
suid  status  name  password
```
Securing role passwords

This unlocks the role:

```sql
alter role doctor_role unlock
```

This sets a new password for the role:

```sql
alter role doctor_role add passwd "dProle1"
```

Running `sp_displayroles` now displays that the role is unlocked and has a password:

```sql
select srid, status, name,
"vers"=substring(password,2,1) from syssrvroles
```

```sql
suid status name vers
------ ------- ------------------- ----- 
33 0 doctor_role 0x05
```
External Authentication

This chapter describes the Adaptive Server features that enable you to authenticate users with authentication data stored in repositories that are external to Adaptive Server.

You can enhance the security for large, heterogeneous applications by authenticating logins with a central repository. Adaptive Server supports these external authentication methods:

- 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.
Configuring Adaptive Server for network-based security

The secure connection between a client and a server can be used for login authentication and message protection.

Figure 5-1: Establishing secure connections between a client and Adaptive Server

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 Adaptive Server.
3. Adaptive Server authenticates the client’s credential with the security mechanism. If the credential is valid, a secure connection is established between the client and Adaptive Server.

If the client requests message protection services:
1. The client uses the security mechanism to prepare the data packet it sends to Adaptive Server.
   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 Adaptive Server.
3 Upon receiving the data packet, Adaptive Server uses the security mechanism to perform any required decryption and validation.

4 Adaptive Server returns results to the client, using the security mechanism to perform the security functions that were requested; for example, Adaptive Server may return the results in encrypted form.

Security services and Adaptive Server

Depending on the security mechanism you choose, Adaptive Server allows you to use one or more of these security services:

- Unified login – authenticates users once, without requiring them to supply a name and password every time they log in to an Adaptive 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 by Adaptive Server.
- 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 Adaptive server to enable secure connection with remote servers. This service is supported by Kerberos security mechanism. Adaptive server currently supports this for connections to remote Adaptive server through CIS.
Configuring Adaptive Server for network-based security

- Remote procedure security – establishes mutual authentication, message confidentiality, and message integrity for remote procedure communications through CIS for Kerberos connections.

Note The security mechanism you are using may not employ all of these services. See “Getting information about available security services” on page 134.

Administering network-based security

Table 5-1 provides an overall process for using the network-based security functions provided by Adaptive Server. You must install Adaptive Server before you can complete the steps in Table 5-1.

Table 5-1: Administering network-based security

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set up configuration files:</td>
<td>Edit the libtcl.cfg file. Edit the objectid.dat file. Edit the interfaces file or Directory Service.</td>
<td>• “Setting up configuration files for security” on page 121 • The Open Client/Server Configuration Guide for your platform</td>
</tr>
<tr>
<td>2. Make sure the security administrator for the security mechanism has created logins for each user and for the Adaptive Server and Backup Server.</td>
<td>The security administrator must add names and passwords for users and servers in the security mechanism.</td>
<td>• The documentation supplied with your security mechanism • “Identifying users and servers to the security mechanism” on page 126</td>
</tr>
<tr>
<td>4. Restart Adaptive Server.</td>
<td>Activates the use security services parameter.</td>
<td>The Configuration Guide for your platform</td>
</tr>
<tr>
<td>5. Add logins to Adaptive Server to support enterprise-wide login.</td>
<td>Use create login to add login accounts. Optionally, specify a default secure login with sp_configure.</td>
<td>“Adding logins to support unified login” on page 130</td>
</tr>
<tr>
<td>6. Enable security mechanism for required remote servers.</td>
<td>Use security mechanism option of sp_serveroption to enable security mechanism for required remote servers.</td>
<td>“Establishing Kerberos security for remote connections” on page 131</td>
</tr>
</tbody>
</table>
Setting up configuration files for security

Configuration files are created during installation at a default location in the Sybase directory structure.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>See</th>
</tr>
</thead>
</table>
| 7. Connect to the server and use security services. | Use `isql_r` or Open Client Client-Library to connect to Adaptive Server, specifying the security services you want to use. | • “Connecting to the server and using the security services” on page 133  
• The Open Client/Server Configuration Guide for your platform  
| 8. Check the security services and security mechanisms that are available. | 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 Adaptive Server, use `select` to query the `syssecmechs` system table. | “Getting information about available security services” on page 134 |

### Table 5-2: Names and locations for configuration files

<table>
<thead>
<tr>
<th>File name</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
</table>
| libtcl.cfg | 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` |
| 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` |

**Note** In Adaptive Server version 12.5.1 and later, you can use a Directory Service instead of the `interfaces` file.
For a detailed description of the configuration files, see the *Open Client/Server Configuration Guide* for your platform.

### Specifying 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 a line for “secmech” that specifies the global identifier or identifiers of the security services you plan to use.

Adaptive Server 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.

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.

*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*.

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*.
Preparation of \textit{libtcl.cfg} to use network-based security

\textit{libtcl.cfg} and \textit{libtcl64.cfg} (for 64-bit applications) contain information about three types of drivers:

\begin{itemize}
  \item Network (Net-Library)
  \item Directory Services
  \item Security
\end{itemize}

A \texttt{driver} is a Sybase 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.

\underline{Entries for network drivers}

The syntax for a network driver entry is:

\texttt{driver=protocol description}

where:

\begin{itemize}
  \item \texttt{driver} – is the name of the network driver.
  \item \texttt{protocol} – is the name of the network protocol.
  \item \texttt{description} – is a description of the entry. This element is optional.
\end{itemize}

\begin{flushright}
\textbf{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.
\end{flushright}

\underline{Entries for Directory Services}

Directory Services entries apply if you want to use a Directory Service instead of the \texttt{interfaces} file. See the configuration documentation for your platform, and the \textit{Open Client/Server Configuration Guide} for your platform.

\underline{Entries for security drivers}

The syntax for a security driver entry is:

\texttt{provider=driver init-string}

where:

\begin{itemize}
  \item \texttt{provider} – is the local name for the security mechanism. The mapping of the local name to a global object identifier is defined in \texttt{objectid.dat}.
  \item The default local names are:
    \begin{itemize}
      \item “csfkrb5” – for the CyberSAFE or MIT Kerberos security mechanism.
      \item “LIBSMSSP” – for Windows LAN Manager on Windows NT or Windows 95 (clients only).
    \end{itemize}
\end{itemize}
If you use a local mechanism name other than the default, change the local name in the objectid.dat file (For an example, see “The objectid.dat file” on page 125).

- **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 Adaptive Server.

After you install Adaptive Server 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=libsyskrb.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. Adaptive Server 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.

The osccfg 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:

```plaintext
[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.

The objectid.dat file

The objectid.dat file maps global object identifiers to local names, such as the one for the Kerberos service (for example, an identifier like 1.3.6.1.4.1.897.4.6.6) to local names, such as “csfkrb5”. The objectid.dat file contains sections such as [CHARSET] for character sets and [SECURITY] for security services. Following is a sample objectid.dat file:

```plaintext
[SECURITY]
secmech
1.3.6.1.4.1.897.4.6.3   = NTLM
1.3.6.1.4.1.897.4.6.6   = 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:

```plaintext
[SECURITY]
csfkrb5=libsybskrb.so secbase=@MYREALM libgss=/krb5/lib/libgss.so
```

to:

```plaintext
[SECURITY]
```
Configuring Adaptive Server for network-based 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
```

**Note** You can specify only one local name per security mechanism.

Identifying 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. See “Using Kerberos” on page 136 for more information about Kerberos and Adaptive Server.

- Windows NT LAN Manager – run the User Manager tool to define users to the Windows NT LAN Manager. Define the Adaptive Server name as a user to Windows NT LAN Manager and display Adaptive Server as that user name.

**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.

Configuring Adaptive Server 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.
Enabling 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, Adaptive Server 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`.

For information about how Adaptive Server determines which security mechanism to use for a particular client, see “Using security mechanisms for the client” on page 134.

Requiring unified login

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
```

Establishing a secure default login

When a user with a valid credential from a security mechanism logs in to Adaptive Server, the server checks whether the user name exists in `master..syslogins`. If it does, Adaptive Server uses that user name. For example, if a user logs in to the Kerberos security mechanism as “ralph,” and “ralph” is in `master..syslogins`, Adaptive Server uses all roles and authorizations defined for “ralph” in the server.

However, if a user with a valid credential logs in to Adaptive Server, but is unknown to the server, the login is accepted only if a secure default login is defined with `sp_configure`. Adaptive Server 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 in Adaptive Server:
   
   ```
   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
   ```
   
   Adaptive Server uses this login for a user who is preauthenticated by a security mechanism but is unknown to Adaptive Server.

   **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.

See “Creating login accounts” on page 17.

### Mapping security mechanism login names to server names

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. All login names in Adaptive Server must be valid identifiers. See Chapter 3, “Expressions, Identifiers, and Wildcard Characters,” in the Reference Manual.

Table 5-3 shows how Adaptive Server converts invalid characters in login names:
Requiring message confidentiality with encryption

To require all messages into and out of Adaptive Server 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
```
Requiring data integrity

Adaptive Server 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 the security mechanism supports it.

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 Adaptive 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 Adaptive Server requires 75,000 2K blocks of memory, including the increased memory for network-based security, execute:

```
sp_configure "max total_memory", 75000
```


Adding logins to support unified login

When users log in to Adaptive Server with a preauthenticated credential, Adaptive Server:

1. Checks whether the user is a valid user in `master..syslogins`. If the user is listed in `master..syslogins`, Adaptive Server accepts the login without requiring a password.
2. If the user name is not in `master..syslogins`, Adaptive Server checks 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.
Therefore, consider whether you want to allow only those users who are defined as valid logins to use Adaptive Server, or whether you want users to be able to log in with the default login. To define the default, add the default login in `master..syslogins` and use `sp_configure`. See “Establishing a secure default login” on page 127.

**General procedure for adding logins**

Follow the general procedure described in Table 5-4 to add logins to the server and, optionally, to add users with appropriate roles and authorizations to one or more databases.

**Table 5-4: Adding logins and authorizing database access**

<table>
<thead>
<tr>
<th>Task</th>
<th>Required role</th>
<th>Command or procedure</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Add a login for the user.</td>
<td>System security officer</td>
<td><code>create login</code></td>
<td>“Creating login accounts” on page 17</td>
</tr>
<tr>
<td>2. Add the user to one or more databases.</td>
<td>System administrator or Database owner</td>
<td><code>sp_adduser</code> – execute this procedure from within the database.</td>
<td>“Adding users to databases” on page 67</td>
</tr>
</tbody>
</table>
| 3. Add the user to a group in a database. | System administrator or Database owner | `sp_changegroup` – execute this procedure from within the database. | • “Changing a user’s group membership” on page 72  
• `sp_changegroup` in the Reference Manual |
| 4. Grant system roles to the user. | System administrator or System security officer | `grant role` | • “Granting and revoking roles” on page 107  
• `grant` in the Reference Manual |
| 5. Create user-defined roles and grant the roles to users. | System security officer | `create role`  
`grant role` | • “Creating and assigning roles to users” on page 91  
• `grant` in the Reference Manual  
• `create role` in the Reference Manual |
| 6. Grant access to database objects. | Database object owners | | Chapter 6, “Managing User Permissions” |

**Establishing Kerberos security for remote connections**

Adaptive Server 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 Adaptive server 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.

Adaptive server 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 Adaptive server can request a Remote Procedure Call (RPC) to Adaptive Server, and for general distributed query processing requests to a remote Adapter Server through CIS by using the Kerberos credential delegation feature. The Kerberos authentication feature used for connections to remote Adaptive 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 Chapter 2, “Understanding Component Integration Services” in the Component Integration Services User Guide.

**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 using 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.
Getting 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
```

Connecting to the server and using the security services

The `isql` and `bcp` utilities include the following command line options to enable network-based security services on the connection:

- `-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.
Configuring Adaptive Server for network-based security

- \( r \) – enables data replay detection.
- \( q \) – enables out-of-sequence detection.
- \(-Z \text{security}_\text{mechanism}\) – specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the \textit{libtcl.cfg} configuration file. If no \textit{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 Adaptive Server, you can define security properties before connecting to the server. For example, to check message sequencing, set the \textit{CS_SEC_DETECTSEQ} property. For information about using security services with Client-Library, see the Open Client Client-Library/C Reference Manual.

**Using security mechanisms for the client**

Adaptive Server, when it is started, determines the set of security mechanisms it supports. See “Determining supported security services and mechanisms” on page 134. From the list of supported security mechanisms, Adaptive Server 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 \textit{isql}), Adaptive Server uses that security mechanism. Otherwise, it uses the first security mechanism listed in the \textit{libtcl.cfg} file.

**Getting information about available security services**

Adaptive Server lets you determine:

- What security mechanisms and services are supported by Adaptive Server
- What security services are active for the current session
- Whether a particular security service is enabled for the session

**Determining supported security services and mechanisms**

A system table, \textit{syssecmechs}, provides information about the security mechanisms and security services supported by Adaptive Server. 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 by Adaptive Server, run:

```sql
select * from syssecmechs
```

### Determining active security services

To determine which security services are active for the current session, use the function `show_sec_services`:

```sql
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:

```sql
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:

```sql
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.
Using 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.

Adaptive Server supports Kerberos through:
- CyberSafe Kerberos libraries
- MIT Kerberos libraries, version 1.3.1
- Native libraries

**Note** To enable Kerberos security options, you must have ASE_SECDIR, the “Security and directory services” package.

Kerberos compatibility

Table 5-5 shows which variation of Kerberos is supported on which platforms.

**Table 5-5: Adaptive Server Kerberos interoperability**

<table>
<thead>
<tr>
<th>Hardware platforms</th>
<th>KDC server</th>
<th>Generic security standard (GSS) client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris 32</td>
<td>CSF, AD, MIT</td>
<td>CSF, MIT, Native</td>
</tr>
<tr>
<td>Solaris 64</td>
<td>CSF, AD, MIT</td>
<td>CSF, MIT, Native</td>
</tr>
<tr>
<td>Linux 32</td>
<td>CSF, AD, MIT</td>
<td>MIT, Native</td>
</tr>
<tr>
<td>Windows 32</td>
<td>CSF, AD</td>
<td>CSF</td>
</tr>
<tr>
<td>AIX 32</td>
<td>CSF</td>
<td>CSF</td>
</tr>
</tbody>
</table>

Use the following keys to read the interoperability matrix:
- **CSF** – CyberSafe Ltd.
- **AD** – Microsoft Active Directory
- **MIT** – MIT version 1.3.1
Starting Adaptive Server under Kerberos

To start Adaptive Server under Kerberos, add the Adaptive 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.
```

**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 Adaptive Server key, instead of prompting for a password, which is useful for writing shell scripts.

Configuring Kerberos

The configuration process is similar, regardless of which variety of Kerberos you use.

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 Adaptive Server will run. The following client packages have been verified to work with:
      - CyberSafe TrustBroker 4.0
      - MIT Kerberos version 1.3.1
   b. Install the Kerberos KDC server on a separate, dedicated machine.

   **Note** KDCs from CyberSafe TrustBroker 4.0, MIT Kerberos v.1.3.1, and Microsoft Windows Active Directory have been verified for use with Adaptive Server.
Configuring Adaptive Server for network-based security

1. 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.

   **Note** Execute the remainder of these steps on the Kerberos client machine.

2. Add Kerberos principal for Adaptive Server `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:

   ```bash
   -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 Adaptive Server principals. Use the Active Directory Users and Computers wizard to guide you through creating users and principals.
Extracting the keytab file for use with Adaptive Server 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 Adaptive Server is located with the CyberSafe program files. For example, c:\Program Files\CyberSafe\v5srvtab is the expected location of the Adaptive Server 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 Adaptive Server and use isql to log in as “sa”. The following steps configure Adaptive Server parameters to use Kerberos security services, and create the user login account. These are the same on both Windows or UNIX machines:

- Change configuration parameter use security services to 1:
  \[sp_configure 'use security services', 1\]
- Add a new login for user, “sybuser1” and then add the user:
  \[create login sybuser1 with password password\]

6 Shut down Adaptive 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
Configuring Adaptive Server for network-based security

- 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 seccomprof=MyRealM libgss=/krb5/lib/libgss.so
  
  A 64-bit CyberSafe Kerberos client library entry follows:
  
  [SECURITY]
  csfkrb5=libsybskrb64.so seccomprof=MyRealM libgss=/appsec-rt
  /lib/64/libgss.so
  
  For a machine that uses MIT Kerberos client libraries, the entry looks
  something like:
  
  [SECURITY]
  csfkrb5=libsybskrb.so seccomprof=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 seccomprof=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 seccomprof=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 CSFC5KTNNAME environment variable on CyberSafe UNIX platforms to specify the keytab file:

```bash
% setenv CSFC5KTNNAME /krb5/v5srvtab
```

For MIT Kerberos, the equivalent environment variable is KRB5_KTNNAME.

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:

```bash
% set path = ( /krb5/lib $path )
```

8 Restart Adaptive Server. You should see:

```
00:00:00:00:00:2001/07/25 11:43:09.91 server
Successfully initialized the security mechanism 'csfkrb5'. The SQL Server will support use of this security mechanism.
```

9 Use isql as UNIX user “sybuser1” (without the -U and -P arguments) to connect:

```bash
% $SYBASE/$SYBASE_OCS/bin/isql -Sase120srv -V
1>...
```

You can also use the encryption option:

```bash
$SYBASE/$SYBASE_OCS/bin/isql -Sase120srv -Vc
```
Using 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 of Adaptive Server running, you must have different principal names for each Adaptive Server.

Specifying the Adaptive Server principal name

Use the DSLISTEN and DSQUERY environment variables, or the dataserver -s<server_name> command line option to specify the Adaptive 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 Adaptive Server. To specify a different name, set SYBASE_PRINCIPAL before starting Adaptive Server to use Kerberos:

```
setenv SYBASE_PRINCIPAL <name of principal>
```

Once you have set an Adaptive Server principal name, Adaptive Server uses the value of this variable to authenticate itself to Kerberos.

To specify an Adaptive Server principal name when starting Adaptive Server, use:

```
-k <server principal name>
```

When you start an Adaptive Server with the Kerberos security mechanism enabled, Adaptive Server first uses the principal name specified with the -k option for Kerberos authentication. If the -k option is not specified, Adaptive Server looks for the principal name in the environment variable SYBASE_PRINCIPAL. If neither is specified, Adaptive Server uses the server name for authentication.

Adaptive Server accepts Kerberos Open Client connections that use different server principal names 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 Adaptive Server name is “secure_ase” and the realm name is “MYREALM.COM.” The Adaptive 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 Adaptive Server principal name is “secure_ase@MYREALM.COM.” If the principal name defined in the Adaptive Server keytab file is “aseprincipal@MYREALM.COM.” you can override the default Adaptive Server 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 Adaptive Server 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 Adaptive Server 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 Adaptive Server principal name used to authenticate with Kerberos is “secure_ase@MYREALM.COM.”

Using sybmapname to handle user principal names

sybmapname converts external user principal names used in the Kerberos environment to the namespace of Adaptive Server 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 Adaptive Server output buffer.
Use the `sybmapname` shared object to perform the custom mapping between the user principal name and the Adaptive Server 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.

```c
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 Adaptive Server error log.

For example, to authenticate a Kerberos user on Adaptive Server:


   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`” on page 146.

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 SLD_LIBRARY_PATH on UNIX machines, and PATH variable on Windows machines. The file should have read and execute permissions for the “sybase” user.

**Note** Sybase recommends that only the “sybase” user is allowed read and execute permissions, and that all other access should be denied.

To verify your login to Adaptive Server using Kerberos authentication, assume that:

- `$SYBASE` refers to your release and installation directory.
- `$SYBASE_ASE` refers to the Adaptive Server version directory that contains your server binary.
- `$SYBASE_OCS` refers to the Open Client/Server version directory.

**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 2** If the client principal name is `user`, and the corresponding entry in `syslogins` table is `USER`, then `sybmapname` can be coded to accept the input string `user` and convert this string to uppercase string `USER`.

`sybmapname` is loaded by Adaptive Server at runtime and uses its logic to do the necessary mapping.

The following actions and output illustrate the `sybmapname` function described in Example 2. 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 Adaptive Server with the Kerberos security mechanism enabled.

To initialize the Ticket Granted Ticket (TGT), which is an 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
```
Configuring Adaptive Server for network-based security

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 Adaptive Server:

$ $SYBASE/$SYBASE_OCS/bin/isql -V -I'pwd'/interfaces

1>

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 Adaptive Server engine.

  **Note** Sybase bears no responsibility for coding errors in `sybmapname`.

• Code defensively, check all pointers before no longer referencing 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.

## Concurrent Kerberos authentication

Adaptive Server version 15.0.3 supports concurrent Kerberos authentication, whereas earlier versions used locking mechanisms during Kerberos authentication to protect internal data structures.

When there are concurrent logins using Kerberos authentication, Adaptive Server now establishes multiple Kerberos authentication sessions.

Version 15.0.3 also resolves an issue with concurrent login sessions, which may be blocked during Kerberos authentication. This concurrency issue occurs when you use prior versions of Adaptive Server with MIT version 1.3.x and 1.4.x Kerberos GSSAPI libraries.
Configuring Adaptive Server for LDAP user authentication

The LDAP user authentication allows client applications to send user name and password information to Adaptive Server for authentication by the LDAP server instead of syslogins. Authentication using the LDAP server allows you to use server-wide passwords instead of Adaptive Server 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 Adaptive Server version 12.5.1 or later, or,
- Searched DN for authentication, available for Adaptive Server 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. Adaptive Server 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.
CHAPTER 5  External Authentication

- 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).

Composed DN algorithm

This is the login sequence when you use the composed DN algorithm:

1. Open Client connects to an Adaptive Server listener port.
2. The Adaptive Server listener accepts the connection.
3. Open Client sends an internal login record.
5. Adaptive Server 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 Adaptive Server sends the search request to the LDAP server.
8. The LDAP server returns the results of the search.
9. Adaptive Server accepts or rejects the login, based on the search results.

Searched DN algorithm

This is the login sequence when you use the searched DN algorithm:

1. Open Client connects to an Adaptive Server listener port.
2. The Adaptive Server listener accepts the connection.
3. Open Client sends an internal login record.
5 Adaptive Server 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 Adaptive Server to reuse connections to DN searches.

6 The LDAP server authenticates the user, returning either a success or failure message.

7 Adaptive Server 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 Adaptive Server reads the results to obtain an a value of attribute from the DN lookup URL.

10 Adaptive Server 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, Adaptive Server sends the search request to the LDAP server.

13 The LDAP server returns the results of the search.

14 Adaptive Server accepts or rejects the login, based on the search results. Adaptive Server 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.

**Configuring LDAP**

You can configure Adaptive Server for LDAP authentication and migrate existing Adaptive Servers to LDAP.

❖ **Configuring LDAP in new Adaptive Server installations**

1 Specify the Adaptive Server 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 to Adaptive Server using `create login`. You can also use `sp_maplogin` to automatically create login accounts upon authentication or apply other login controls.

❖ **Migrating existing Adaptive Servers to LDAP**

To avoid disruption of service in existing server installations, migrate Adaptive Server to LDAP.

1 Specify an LDAP URL search string to Adaptive Server.

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.

### 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`.

See the *Reference Manual: Commands*.

<table>
<thead>
<tr>
<th>Composed DN examples</th>
</tr>
</thead>
</table>

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 Adaptive Server 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.
Configuring Adaptive Server for LDAP user authentication

- The Adaptive Server 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, Adaptive Server replaces the wildcard with the Adaptive Server 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=mycompany,dc=com

- After a successful bind operation, Adaptive Server uses the connection 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=*'

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.

  a  Set the access account information:

  sp_ldapadmin set_access_acct, 'cn=Admin Account, cn=Users, dc=mycompany, dc=com', 'Admin Account secret password'

  b  Set the primary URL:

  sp_ldapadmin set_primary_url, 'ldap://hostname:389/

  c  Set the DN lookup URL search string:
CHAPTER 5    External Authentication

sp_ldapadmin set_dn_lookup_url,
'ldap://hostname:389/cn=Users,dc=mycompany,dc=com?distinguishedName
?one?samaccountname=*'

On Windows 2000, 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 Adaptive Server 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 Adaptive
Server 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))'

Adaptive Server 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))'

LDAP user authentication password information changes

There are two LDAP user authentication-related informational messages that
Adaptive Server obtains from the LDAP server and passes to the client:
• If you log in to an Adaptive 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 Adaptive Server 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.

Note Configure your LDAP server to give this additional information. Additionally, Adaptive Server must support the transmission of LDAP password controls to an LDAP client.

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, Adaptive Server attempts to connect to the secondary LDAP directory server specified by the secondary URL. Adaptive Server 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 Adaptive Server to attempt to fail over to a secondary URL. Failures returned by LDAP bind or search operations do not cause Adaptive Server to fail over to the secondary URL.
Adaptive Server 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 Adaptive Server. Alternatively, the database administrator can choose administration options that allow flexibility with login accounts when integrating Adaptive Server with external authentication mechanisms such as LDAP server. The database administrator continues to administer the Adaptive Server account roles, default database, default language, and other login-specific attributes using traditional commands and procedures.

Table 5-6 describes the updates to syslogins table Adaptive Server makes at login time. 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.

Table 5-6: Updates to syslogins from LDAP

<table>
<thead>
<tr>
<th>Does the row exist in syslogins?</th>
<th>LDAP server authentication succeeds?</th>
<th>Changes in syslogins</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>No change, login fails</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No change, login fails</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Update row if password has changed</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No change</td>
</tr>
</tbody>
</table>

Secondary lookup server support

Adaptive Server provides uninterrupted support to Adaptive Server 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 Adaptive Server administrative commands.
- READY – indicates that the URL is ready to accept connections.
- ACTIVE – indicates that the URL has performed a successful LDAP user authentication.
Configuring Adaptive Server for 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 Adaptive Server administrator to READY.
6. New connections are authenticated using the primary server.

Note: Once Adaptive 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 Adaptive Server encounters errors connecting to an LDAP server, it retries the authentication three times. If the errors persist, the LDAP server is marked as FAILED. See “Troubleshooting LDAP user authentication errors” on page 163 for information on the LDAP errors that force Adaptive Server into a retry loop.

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]

Adaptive Server version 12.5.4 and later includes the following sp_ldapadmin options that support secondary servers.

- 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

**LDAP server state transitions**

Table 5-7 – Table 5-12 list LDAP server state transitions when each sp_ldapadmin commands is executed.

Table 5-7 shows the state transitions when you execute sp_ldapadmin set_URL, where set_URL represents one of these commands:

- set_dn_lookup_url
- set_primary_url
- set_secondary_dn_lookup_url
- set_secondary_url

*Table 5-7: State transitions when sp_ldapadmin set_URL is executed*

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>RESET</td>
</tr>
<tr>
<td>RESET</td>
<td>RESET</td>
</tr>
<tr>
<td>READY</td>
<td>READY</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>RESET</td>
</tr>
<tr>
<td>FAILED</td>
<td>RESET</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>RESET</td>
</tr>
</tbody>
</table>

Table 5-8 shows the state transitions when you execute sp_ldapadmin suspend.

*Table 5-8: State transitions when sp_ldapadmin suspend is executed*

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>Error</td>
</tr>
<tr>
<td>RESET</td>
<td>SUSPENDED</td>
</tr>
<tr>
<td>READY</td>
<td>SUSPENDED</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>SUSPENDED</td>
</tr>
<tr>
<td>FAILED</td>
<td>SUSPENDED</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>SUSPENDED</td>
</tr>
</tbody>
</table>

Table 5-9 shows the state transitions when you execute sp_ldapadmin activate.

*Table 5-9: State transitions when sp_ldapadmin activate is executed*

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>Error</td>
</tr>
<tr>
<td>RESET</td>
<td>READY</td>
</tr>
<tr>
<td>READY</td>
<td>READY</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>FAILED</td>
<td>READY</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>READY</td>
</tr>
</tbody>
</table>

The following tables show the LDAP server state transitions carried out implicitly by Adaptive Server.

Table 5-10 shows the state transitions when Adaptive Server is restarted:

*Table 5-10: State transitions when Adaptive Server is restarted*

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>INITIAL</td>
<td>INITIAL</td>
</tr>
<tr>
<td>RESET</td>
<td>RESET</td>
</tr>
</tbody>
</table>
Adaptive Server only attempts an LDAP login if the LDAP server is in a READY or ACTIVE state. Table 5-11 shows the state transitions:

**Table 5-11: State transitions when an LDAP login succeeds**

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>READY</td>
<td>READY</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>READY</td>
</tr>
<tr>
<td>FAILED</td>
<td>FAILED</td>
</tr>
<tr>
<td>SUSPENDED</td>
<td>SUSPENDED</td>
</tr>
</tbody>
</table>

Table 5-12 shows the state transitions when an LDAP login fails:

**Table 5-12: State transitions when an LDAP login fails**

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
</tr>
</thead>
<tbody>
<tr>
<td>READY</td>
<td>FAILED</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>FAILED</td>
</tr>
</tbody>
</table>

**LDAP user authentication tuning**

Configure and tune Adaptive Server options based on the load of incoming connections and the Adaptive Server-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 Adaptive Server/LDAP server infrastructure.

Configure the `set_abandon_ldapua_when_full` option to specify Adaptive Server 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 Adaptive Server is processing.

- `set_num_retries` – sets the number of attempts. Tune this number according to the number of transient errors between Adaptive Server 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 Adaptive Server 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.

Adding tighter controls on login mapping

Use `sp_maplogin` to map users that are authenticated with LDAP or PAM to the local Adaptive Server login.

**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`.

Adaptive Server 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 Adaptive Server 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.

**Examples**

**Example 1**  Maps an LDAP user to the Adaptive Server “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 Adaptive Servers. Auditing is enabled, and login events are recorded in the audit trail.

To map these administrator accounts to “sa,” enter:

```sql
sp_maplogin LDAP, 'adminA', 'sa'
go
sp_maplogin LDAP, 'adminB', 'sa'
go
```

Require all users to authenticate using LDAP authentication:

```sql
sp_configure 'enable ldap user auth', 2

go
```

When “adminA” authenticates during login to Adaptive 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 Adaptive Servers being managed.

This example also allows different external identities and passwords to be used for authentication, while their actions within Adaptive Server still require the special privileges associated with “sa” account.
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 Adaptive Server to handle requests on behalf of users of the middle-tier application.

Configure Adaptive Server for both LDAP and PAM user authentication:

```sql
sp_configure 'enable ldap user auth', 2  
go
sp_configure 'enable pam user auth', 2
  go
```

Establish an Adaptive Server login `appX` locally with permissions that are appropriate for the middle-tier application:

```sql
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 Adaptive 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,”

```sql
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 Adaptive Server 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.”
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 Adaptive Server login, and if the mapping is successfully authenticated, Adaptive Server updates the internal login password to match the external user’s password. For example:

1. A user has an Adaptive Server login name of user_ase (with password user_password), and an LDAP login name of user_ldap (with password user_ldappassword).

   The produces a one to one mapping for user_ldap to user_ase.

2. When user_ldap logs into Adaptive Server using the user_ldappassword, Adaptive Server updates the password for user_ase to user_ldappassword.

The benefit of mapping the Adaptive Server 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 Adaptive Server authentication, the user appears to have uninterrupted authentication to Adaptive Server because the password is updated locally when it is used to authenticate the login.

However, Adaptive Server does not update the password locally when more than one external user is mapped to the local user. If the LDAP server crashes, Adaptive Server cannot authenticate multiple external users mapped to a single Adaptive Server user.

Troubleshooting LDAP user authentication errors

Adaptive Server 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, Adaptive Server 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.
Configuring Adaptive Server for LDAP user authentication

- 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.

Configuring an LDAP server


Configure a connection to an LDAP server

1. Make sure that all trusted root certificates are located in the same file.
After you define the trusted servers, Adaptive Server configures a secure connection, where *servername* is the name of the current Adaptive Server. If you:

- Have defined `$SYBASE_CERTDIR`, Adaptive Server loads certificates from `$SYBASE_CERTDIR/servername.txt` (for UNIX) or `%SYBASE_CERTDIR%\servername.txt` (for Windows).
- Have not defined `$SYBASE_CERTDIR`, Adaptive Server loads certificates from `$SYBASE/$SYBASE_ASE/certificates/servername.txt` (for UNIX) or `%SYBASE%\%SYBASE_ASE%\certificates\servername.txt` (for Windows).

2 Restart Adaptive 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}
   ```

**Note** LDAP server connections do not have a connect timeout option; if the LDAP server stops responding, all login connections also stop responding.

### LDAPS user authentication enhancements

In earlier versions of Adaptive Server, if you modify the Certifying Authority (CA) trusted root file, you must restart Adaptive Server for the modifications to take effect. Adaptive Server version 15.0.3 and later supports modifications to the trusted root file, so that restarting the server is unnecessary. A new subcommand, `reinit_descriptors`, which unbinds the LDAP server descriptors and reinitializes the user authentication subsystem. For the syntax of this option see *Reference Manual: Procedures*.

- This command requires System Security Officer permissions.
Automatic LDAP user authentication and failback

Adaptive Server 15.0.3 provides support for a secondary LDAP server. Previously, after bringing a failed primary LDAP server online, it was necessary to activate the LDAP server manually, in order to authenticate new LDAP logins and move them to the primary LDAP server.

In versions 15.0.3 and later, a new chore has been added to Adaptive Server’s housekeeping utility to activate an LDAP server automatically: 'set_failback_interval’ – for syntax, see “Setting the LDAP failback time interval” on page 167.

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.

• If the 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.
Setting the LDAP failback time interval

The syntax for `sp_ldapadmin set_failback_interval` is the following, where `time_inMinutes` is the value from -1 to 1440 minutes (24 hours):

```
sp_ldapadmin 'set_failback_interval', time_inMinutes
```

- 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)
```

Examples

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'.

Configuring Adaptive Server 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 Adaptive Server 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.

Note PAM support is currently available on Linux and on Solaris platforms.
For more information on PAM user authentication, see your operating system
documentation.
Adaptive Server 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.

### Enabling PAM in Adaptive Server

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.

**Note** PAM modules you create should comply with RFC 86.0 “Unified Login With Pluggable Authentication Modules (PAM).” Adaptive Server supports the authentication management module of the RFC. It does not support the account management, session management, or password management modules.
Configuring operating systems

To enable PAM support, configure your operating system as follows:

- 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.

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.

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
```

**Note** $ISA is the only variable allowed in `pam.conf`.

**Configuring Adaptive Server for PAM user authentication**

`enable pam user auth` enables PAM user authentication support:

````
sp_configure "enable pam user auth", 0 | 1 | 2
```

where:

- **0** – disables PAM authentication. This is the default.
• 1 – indicates Adaptive Server first attempts PAM authentication, and then uses syslogins authentication if PAM authentication fails.
• 2 – indicates only PAM authentication may be used.

**Note** When PAM is enabled, password management is delegated to the PAM service providers.

---

**Adaptive Server logins and PAM user accounts**

After you have set `enable PAM user authentication` and completed the PAM configuration for both Adaptive Server 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 Adaptive Server. Alternatively, the database administrator can choose administration options that allow flexibility with login accounts when integrating Adaptive Server with external authentication mechanisms such as PAM. The database administrator continues to administer the Adaptive Server account roles, default database, default language, and other login-specific attributes using traditional commands and procedures.

Table 5-13 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`. 

Enhanced login controls

Configure Adaptive Server to allow the server-wide authentication mechanism according to the methods discussed in the LDAP and PAM sections earlier. You can also configure Adaptive Server to specify the authentication mechanism for each individual login on the server using Adaptive Server enhanced login controls described below.

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.

Forcing authentication

You can force a login to use a specific authentication process by using these parameters for `alter login` and `create login`:

- ASE – use Adaptive Server 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 Adaptive Server checks if there is any external authentication mechanism defined, and if there is, it is used. Otherwise, it uses Adaptive Server authentication.

Adaptive Server checks for external authentication mechanisms in the following order:

### Table 5-13: Updates to syslogins from PAM

<table>
<thead>
<tr>
<th>Does the row exist in syslogins?</th>
<th>PAM authentication succeeds?</th>
<th>Changes in syslogins</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>No change, login fails</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No change, login fails</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Update row if password has changed</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No change</td>
</tr>
</tbody>
</table>

Adaptive Server Enterprise
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, Adaptive Server 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
```

**Mapping logins using sp_maplogin**

Use `sp_maplogin` to map logins:

```
sp_maplogin (authentication_mech | null),
(client_username | null), (action | login_name | null)
```

where:

- `authentication_mech` – is one of the valid values specified for the authenticate with option in `sp_maplogin`.
- `client_username` – is an external user name, which can be an operating system name, a user name for an LDAP server, or anything else the PAM library understands. A null value indicates that any login name is valid.
- `action` – indicates `create login` or `drop`. When you use `create login`, the login is created as soon as is authenticated. Use `drop` to remove logins.
Enhanced login controls

- `login_name` is an Adaptive Server login that already exists in `syslogins`. This example maps external user “jsmith” to the Adaptive Server user “guest.” Once authenticated, “jsmith” has the privileges of “guest.” The audit login record shows both the `client_username` and the Adaptive Server user name:

  ```
  sp_maplogin NULL, "jsmith", "guest"
  ```

  This example tells Adaptive Server to create a new login for all external users authenticated with LDAP, if a login does not already exist:

  ```
  sp_maplogin LDAP, NULL, "create login"
  ```

Displaying mapping information

`sp_helpmaplogin` displays mapping information:

```
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 Adaptive Server. 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

<table>
<thead>
<tr>
<th>authentication</th>
<th>client name</th>
<th>login name</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>jsmith</td>
<td>guest</td>
</tr>
<tr>
<td>LDAP</td>
<td>NULL</td>
<td>create login</td>
</tr>
</tbody>
</table>
```

Determining the authentication mechanism

Use the `@@authmech` global variable to determine the authentication mechanism Adaptive Server uses.

For example, if Adaptive Server 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, Adaptive Server attempts to authenticate Joe, using LDAP user authentication. If Joe fails authentication as a user in LDAP, Adaptive Server authenticates Joe using Adaptive Server authentication, and if that succeeds, he logs in successfully.
@@authmech global has this value:

```
select @@authmech
---------------------
ase
```

If Adaptive Server 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
```
Enhanced login controls
CHAPTER 6
Managing User Permissions

This chapter describes the use and implementation of user permissions.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>177</td>
</tr>
<tr>
<td>Permissions for creating databases</td>
<td>179</td>
</tr>
<tr>
<td>Database owner privileges</td>
<td>180</td>
</tr>
<tr>
<td>Database object owner privileges</td>
<td>182</td>
</tr>
<tr>
<td>Other database user privileges</td>
<td>182</td>
</tr>
<tr>
<td>Permissions on system procedures</td>
<td>183</td>
</tr>
<tr>
<td>Granting and revoking permissions</td>
<td>183</td>
</tr>
<tr>
<td>Acquiring the permissions of another user</td>
<td>193</td>
</tr>
<tr>
<td>Changing database object ownership</td>
<td>199</td>
</tr>
<tr>
<td>Reporting on permissions</td>
<td>203</td>
</tr>
<tr>
<td>Using views and stored procedures as security mechanisms</td>
<td>206</td>
</tr>
<tr>
<td>Executing a procedure with execute as owner or execute as caller</td>
<td>214</td>
</tr>
<tr>
<td>Using row-level access control</td>
<td>221</td>
</tr>
</tbody>
</table>

**Note** Permission requirements for operations mentioned in this chapter assume that granular permissions is disabled. Operations may differ when granular permissions is enabled. See Chapter 8, “Using Granular Permissions,” for more information on granular permissions.

**Overview**

*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.
Adaptive Server’s 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 User Guide for Encrypted Columns). 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.

For details on assuming another user’s identity to acquire permissions on a database or object, see “Acquiring the permissions of another user” on page 193.

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. See “Using views and stored procedures as security mechanisms” on page 206.

Permissions for creating databases

Only a system administrator can grant permission to use the create database command. 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.

To create a database that is to be owned by another user:

1. Issue the create database command in the master database.
2. Switch to the new database with the use command.
3. Execute sp_changedbowner.
Changing 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`.

Sybase 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 Adaptive 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.”

**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.

Database owner privileges

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). The database owner has full privileges to do anything inside that database, and must explicitly grant permissions to other users with the `grant` command.

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
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.

Other database user privileges

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.
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.

Granting and revoking permissions

You can control the following types of permissions with grant and revoke:

- 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.
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 6-1 lists the types of object access permissions and the objects to which they apply.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>Table, view, column</td>
</tr>
<tr>
<td>update</td>
<td>Table, view, column</td>
</tr>
<tr>
<td>insert</td>
<td>Table, view</td>
</tr>
<tr>
<td>delete</td>
<td>Table, view</td>
</tr>
<tr>
<td>references</td>
<td>Table, column</td>
</tr>
<tr>
<td>execute</td>
<td>Stored procedure</td>
</tr>
<tr>
<td>truncate table</td>
<td>Table</td>
</tr>
<tr>
<td>delete statistics</td>
<td>Table</td>
</tr>
<tr>
<td>update statistics</td>
<td>Table</td>
</tr>
<tr>
<td>decrypt</td>
<td>Table, view, column</td>
</tr>
<tr>
<td>select</td>
<td>Encryption key</td>
</tr>
</tbody>
</table>

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. See “Granting Predicated Privileges” on page 251.

Concrete identification

Adaptive Server identifies users during a session by login name. This identification applies to all databases in the server. 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, which allows the server to recognize when permissions on the object can be granted implicitly.

If an Adaptive Server 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:

```sql
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.

Special requirements for SQL92 standard compliance

When you have used the set command to turn ansi_permissions on, additional permissions are required for update and delete statements. Table 6-2 summarizes the required permissions.

<table>
<thead>
<tr>
<th>Permissions required: set ansi_permissions off</th>
<th>Permissions required: set ansi_permissions on</th>
</tr>
</thead>
<tbody>
<tr>
<td>update update permission on columns where values are being set</td>
<td>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</td>
</tr>
</tbody>
</table>
Granting and revoking permissions

<table>
<thead>
<tr>
<th>Permissions required: ansi_permissions off</th>
<th>Permissions required: ansi_permissions on</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete delete permission on the table</td>
<td>delete permission on the table from which rows are being deleted and select permission on all columns appearing in the where clause</td>
</tr>
</tbody>
</table>

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.

Examples of granting object access permissions

This statement gives Mary and the “sales” group permission to insert into and delete from the titles table:

```sql
grant insert, delete
on titles
  to mary, sales
```

This statement gives Harold permission to use the stored procedure makelist:

```sql
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:

```sql
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:

```sql
grant select, update, delete
on authors
  to aubrey
  with grant option
```

This statement grants permission to the payroll employees to update salaries during December:

```sql
grant update (salary)
  on employee
  where date_part(month, getdate()) = 12
  to payroll_role
```
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:

```sql
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:

```sql
revoke update
on authors
from clare
cascade
```

This statement revokes permission from operators to execute the custom stored procedure new_sproc:

```sql
revoke execute
on new_sproc
from oper_role
```

Granting 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.

Server-wide and database-specific dbcc commands

dbcc commands are either:

- Database-specific – dbcc commands that 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.
Granting and revoking permissions

* Server-wide – `dbcc` commands such as `tune` that 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.

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`.

**dbcc grantees and users in databases**

`grant dbcc` and `revoke dbcc` work on users in databases.

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.
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:

```sql
sp_helprotect system_table_name
```

For example, to check the permissions of `sysrvroles` in the master database, execute:

```sql
use master
go
sp_helprotect sysrvroles
go
```

The default situation is that no users—including database owners—can modify the system tables directly. Instead, the T-SQL commands and the system procedures supplied with Adaptive Server modify the system tables. This helps guarantee integrity.

**Warning!** Although Adaptive Server provides a mechanism that allows you to modify system tables, Sybase strongly recommends that you do not do so.

Granting 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 Adaptive Server builds a new database. The partial syntax is:

```sql
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:

<table>
<thead>
<tr>
<th>sysalternates</th>
<th>sysjars</th>
<th>sysquerymatrics</th>
<th>systhresholds</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysattributes</td>
<td>syskeys</td>
<td>sysqueryplans</td>
<td>systypes</td>
</tr>
<tr>
<td>syscolumns</td>
<td>syslogs</td>
<td>sysreferences</td>
<td>sysusermessages</td>
</tr>
</tbody>
</table>
Granting and revoking permissions

<table>
<thead>
<tr>
<th>syscomments</th>
<th>sysobjects</th>
<th>sysroles</th>
<th>sysusers</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysconstraints</td>
<td>syspartitionkeys</td>
<td>syssegments</td>
<td>sysxtypes</td>
</tr>
<tr>
<td>sysdepends</td>
<td>syspartitions</td>
<td>syslices</td>
<td></td>
</tr>
<tr>
<td>sysgames</td>
<td>sysprocedures</td>
<td>sysstatistics</td>
<td></td>
</tr>
<tr>
<td>sysindexes</td>
<td>sysprotects</td>
<td>systabstats</td>
<td></td>
</tr>
</tbody>
</table>

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 sysencryptedkeys from public
- Grants select on all columns of sysencryptedkeys to sso_role

If you run this command from the master database, default permissions for the following system tables are granted or revoked:

<table>
<thead>
<tr>
<th>syscharsets</th>
<th>syslanguages</th>
<th>sysmessages</th>
<th>sysservers</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysconfigures</td>
<td>syslisteners</td>
<td>sysmonitor</td>
<td>syssessions</td>
</tr>
<tr>
<td>syscurconfigs</td>
<td>syslocks</td>
<td>sysprocesses</td>
<td>sysravroles</td>
</tr>
<tr>
<td>sysdatabases</td>
<td>syslogin</td>
<td>sysremotelogins</td>
<td>systimeranges</td>
</tr>
<tr>
<td>sysdevices</td>
<td>sysloginrole</td>
<td>sysresourcelimits</td>
<td>systranactions</td>
</tr>
<tr>
<td>sysengines</td>
<td>syslogshold</td>
<td>sysscmchec</td>
<td>syssuses</td>
</tr>
</tbody>
</table>

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(deftbaud) 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 syslogsins(password) to public
• Revokes select on syslogsins(audflags) from public
• Grants select on syslogsins 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

Combining 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
```
Granting and revoking permissions

```
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.

**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.

Understanding 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:

```
on titles (price, advance)
from public
revoke delete
on titles
from mary, sales, john
```
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.

Grant dbcc and set proxy issue warning for fipsflagger
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.

Acquiring the permissions of another user
Adaptive Server provides two ways 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. See “Using setuser” on page 194.
- proxy authorization allows one user to assume the identity of another user on a server-wide basis. See “Using proxy authorization” on page 195.
Acquiring the permissions of another user

Using setuser

A database owner may 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. Adaptive Server 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.

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*/
```
CHAPTER 6  Managing User Permissions

Using proxy authorization

With the proxy authorization capability of Adaptive Server, 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 Adaptive Server.

Warning! 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.

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.

Using 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.
Acquiring the permissions of another user

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.

Executing proxy authorization

Follow these rules when you execute 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
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
---------------------------------- -------------------
```
Acquiring the permissions of another user

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
```

---

Proxy authorization for applications

Figure 6-1 shows an application server logging in to Adaptive 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.

**Figure 6-1: Applications and proxy authorization**

Tom, Sue, and John establish sessions with the Application Server:

The application server (“appl”) on Adaptive Server executes:

```
set proxy "tom"
```  
(SQL command for Tom)

```
set proxy "sue"
```  
(SQL command for Sue)

```
set proxy "John"
```  
(SQL command for John)
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.

Supported object types

The ownership of the following objects can be transferred from one owner to another. The ownership of objects not listed below cannot be changed.

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

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
Changing database object ownership

- Check constraints
- Reference constraints
- Partition conditions
- Computed columns

Authorization

- 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, 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.

Transferring 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
```

### Transferring ownership of objects in the system database

Use caution to change the ownership of objects in the following system databases that are supplied and managed by Sybase: sybsecurity, sybsystemdb, model, sybsystemprocs, sybsyntax, dbccdb, and tempdb. Do not change the ownership of system objects that are supplied and managed by Sybase, such as but not limited to, user tables with spt_prefix, and system stored procedures with sp_prefix. Changing the ownership of these objects can make the system unusable.

### Transferring 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 `*`.

### Using `preserve permissions`

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.
Changing database object ownership

For example, bill owns table bill.encl_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.encl_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.encl_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.

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.

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.

Transferring 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.
Encryption key copy owners

When using the `alter... modify owner` command, the user who has been assigned a key copy cannot be the new owner of the encryption key.

After the owner of an 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`.

### Reporting on permissions

Table 6-3 lists the system procedures for reporting information about proxies, object creation, and object access permissions:

<table>
<thead>
<tr>
<th>To report information on</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proxies</td>
<td>system tables</td>
</tr>
<tr>
<td>Users and processes</td>
<td><code>sp_who</code></td>
</tr>
<tr>
<td>Permissions on database objects or users</td>
<td><code>sp_procpol</code></td>
</tr>
<tr>
<td>Permissions on specific tables</td>
<td><code>sp_procpol</code></td>
</tr>
</tbody>
</table>

### Querying 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.

You might execute 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
Reporting on permissions

- 2 for revoke

For more information about the sysprotects table, see the Reference Manual: Building Blocks.

Displaying information about users and processes

sp_who displays information about all current Adaptive Server users and processes or about a particular user or process. The results of sp_who include the loginame and origname. 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:

```sql
set proxy susie
sp_who
```

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.

Reporting 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. The syntax is:

```sql
sp_helprotect [name [, username [, "grant" [,"none"]|"granted"|"enabled"|role_name]]]]
```

where:

- `name` – is either the name of the table, view, or stored procedure, or the name of a user, group, or role in the current database. If you do not provide a name, sp_helprotect reports on all permissions in the database.
- `username` – is a user’s name in the current database.

If you specify `username`, only that user’s permissions on the specified object are reported. If `name` is not an object, sp_helprotect checks whether `name` is a user, group, or role and if it is, lists the permissions for the user, group, or role. If you specify the keyword `grant`, and `name` is not an object, sp_helprotect displays all permissions granted by with grant option.

- `grant` – displays the permissions granted to `name` with grant option.
• none – ignores roles granted to the user.
• granted – includes information on all roles granted to the user.
• enabled – includes information on all roles activated by the user.
• role_name – displays permission information for the specified role only, regardless of whether this role has been granted to the user.

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
```

<table>
<thead>
<tr>
<th>grantor</th>
<th>grantee</th>
<th>type</th>
<th>action</th>
<th>object</th>
<th>column</th>
<th>predicate</th>
<th>grantable</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Select</td>
<td>titles</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>advance</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>notes</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>price</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>pub_id</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>pubdate</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>title</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>title_id</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>total_sales</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Update</td>
<td>titles</td>
<td>type</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>grantor</th>
<th>grantee</th>
<th>type</th>
<th>action</th>
<th>object</th>
<th>column</th>
<th>predicate</th>
<th>grantable</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>judy</td>
<td>Grant</td>
<td>Select</td>
<td>publishers</td>
<td>all</td>
<td>NULL</td>
<td>TRUE</td>
</tr>
</tbody>
</table>
Using views and stored procedures as security mechanisms

Reporting 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
```

<table>
<thead>
<tr>
<th>grantor</th>
<th>grantee</th>
<th>type</th>
<th>action</th>
<th>object</th>
<th>column</th>
<th>predicate</th>
<th>grantable</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Delete</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Delete Statistics</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Insert</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>References</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Transfer Table</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Update</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>guest</td>
<td>Grant</td>
<td>Update Statistics</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>dbo</td>
<td>public</td>
<td>Grant</td>
<td>Select</td>
<td>sales</td>
<td>All</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

(1 row affected)

(return status = 0)

For more information about the output of `sp_helprotect`, see the Reference Manual: Procedures.

Using 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” on page 210.

Adaptive Server makes permission checks, as required, when the view or procedure is used. When you create the view or procedure, Adaptive Server makes no permission checks on the underlying objects.
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.

Let’s say 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, and then give all users permission on the view but only the Sales Department permission on the table:

```sql
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

Adaptive Server 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.
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` 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` clause.

The `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`.

See Executing a procedure with `execute as owner` or `execute as caller`. For syntax see, `create procedure` in Reference Manual: Commands.

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:

```sql
create proc test_proc
as
if (has_role("sa_role") = 0)
begin
    print "You don't have the right role"
```
Using views and stored procedures as security mechanisms

```sql
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.

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, Adaptive Server 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, Adaptive Server 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, Adaptive Server checks the permissions for object B. In this way, Adaptive Server allows the owner of the original data 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, Adaptive Server allows Sue 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.

### Example of views and ownership chains

Say Joe creates a view called auview2, which depends on Mary’s view auview1. Joe grants Sue select permission on auview2.

**Figure 6-2: Ownership chains and permission checking for views, case 1**

<table>
<thead>
<tr>
<th>Sue’s permission</th>
<th>Objects</th>
<th>Ownership</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>auview2</td>
<td>Joe</td>
<td>Sue not owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check permissions</td>
</tr>
<tr>
<td>select</td>
<td>auview1</td>
<td>Mary</td>
<td>Different owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check permissions</td>
</tr>
<tr>
<td>none</td>
<td>authors</td>
<td>Mary</td>
<td>Same owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No permission check</td>
</tr>
</tbody>
</table>

Adaptive Server checks the permissions on auview2 and auview1, and finds that Sue can use them. Adaptive Server 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:
Using views and stored procedures as security mechanisms

**Figure 6-3: Ownership chains and permission checking for views, case 2**

<table>
<thead>
<tr>
<th>Sue’s permission</th>
<th>Objects</th>
<th>Ownership</th>
<th>Checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>select</td>
<td>auview3</td>
<td>Mary</td>
<td>Sue not owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check permissions</td>
</tr>
<tr>
<td>select</td>
<td>auview2</td>
<td>Joe</td>
<td>Different owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check permissions</td>
</tr>
<tr>
<td>select</td>
<td>auview1</td>
<td>Mary</td>
<td>Different owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Check permissions</td>
</tr>
<tr>
<td>none</td>
<td>authors</td>
<td>Mary</td>
<td>Same owner</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>No permission check</td>
</tr>
</tbody>
</table>

When Sue tries to access auview3, Adaptive Server 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, Adaptive Server allows the access. Adaptive Server 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.

**Example of procedures and ownership chains**

Procedures follow the same rules as views. For example, suppose the ownership chain looks like this:
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.

Adaptive Server checks Sue’s permissions on `proc4` and all objects it references each time she executes `proc4`. Adaptive Server 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, Adaptive Server 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.
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 Adaptive Server 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.


Executing a procedure with execute as owner or execute as caller

In versions of Adaptive Server 15.7 ESD #2, 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, Adaptive Server 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 Adaptive Server 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, Adaptive Server will look up a table of that name belonging to the procedure owner. If no such table exists, Adaptive Server 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.
Executing a procedure with execute as owner or execute as caller

- All access control checks are based on the procedure owner’s permission, his group, his system roles, his default user-defined roles, and 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_proxmode.

- 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 of statements executed within the procedure show the procedure owner’s name and the option execute as owner.

- 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, Adaptive Server will look up a table of that name belonging to the user who called the procedure. If no such table exists, Adaptive Server 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_proxmode.

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 of statements executed within the procedure show the procedure caller's name and execute as caller clause.

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:

```sql
create procedure p_emp
    select * into emp_interim
    from jane.employee
grant execute on p_emp to bill
```

Bill executes the procedure:

```sql
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 him 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:

```sql
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.
• `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:

```sql
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. See “Granting Predicated Privileges” on page 251 for more information.
• Bill must have `create table` permission. `emp_interim` is created on behalf of Bill, meaning Bill is the owner.
```

### Creating a procedure with references to an object with an unqualified name

In the following example, the procedure has no `execute as` clause:

```sql
create procedure insert p
  insert t1 (c1) values (100)
grant execute on insert p to bill

Bill executes the procedure:

```sql
exec jane.insert p

• Adaptive Server will look for a table named `t1` owned by Jane. If `jane.t1` does not exist, Adaptive Server will look for `dbo.t1`.```
If Adaptive Server resolves \( t1 \) to \( dbo.t1 \), Bill must have permission to insert into \( t1 \).

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 \texttt{execute as owner}:

```sql
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:

```sql
exec jane.insert p
```

Adaptive Server will look for a table named \( t1 \) owned by Jane. If \( jane.t1 \) does not exist Adaptive Server will look for \( dbo.t1 \).

If Adaptive Server resolves \( t1 \) 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 \texttt{execute as caller}:

```sql
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:

```sql
exec jane.insert p
```

Adaptive Server will look for a table named \( t1 \) owned by Bill. If \( bill.t1 \) does not exist Adaptive Server will look for \( dbo.t1 \).

If Adaptive Server resolves \( t1 \) to \( dbo.t1 \), Bill must have permission to insert into \( t1 \).
Procedures that invoke a nested procedure in another database with a fully qualified name

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:

```sql
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:

- Adaptive Server 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:

```sql
create procedure p master
    with execute as caller
    as exec otherdb.jim.p_child
grant execute p master to bill
```

Bill executes the procedure:

```sql
exec jane.p_master
```

- Adaptive Server 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.
Using row-level access control

Row-level access control enables the database owner or table owner to create a secure data access environment automatically, by providing:

- 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

Adaptive Server 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 Adaptive Server uses, and you need an ASE_RLAC license option. Row-level access control is a dynamic option, so you need not restart Adaptive 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. See “Granting Predicated Privileges” on page 251.

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. Adaptive Server 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, Adaptive Server enforces the domain rule on the table that is updated, and the access rule on all tables that are read.

For example:

```sql
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`, Adaptive Server 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`. Adaptive Server 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. Adaptive Server enforces the access rule on any tables in your application that have columns of type `username`.

Application developers can write flexible access rules using Java and application contexts, described in “Access rules as user-defined Java functions” on page 227, and “Using the Application Context Facility” on page 230.

### Syntax for access rules

Use the `access` parameter in the `create rule` syntax to create access rules.

```sql
create [or|and] access rule (access_rule_name) as (condition)
```

### Creating a sample table with access rules

This section shows the process of creating a table and binding an access rule to it.
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

Adaptive Server 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, Adaptive Server 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:
Using row-level access control

```
drop rule "T.username"
```

**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.

**Using access and extended access rules**

**Create access rules**

The following steps create access rules:

```sql
create access rule empid1_access
as @empid = 1
create access rule deptno1_access
as @deptid = 2
```

The following steps create OR access rules:

```sql
create or access rule name1_access
as @name = "smith"
create or access rule phone_access
as @phone = "9999"
```

**Create table**

This step creates a test table:

```sql
create table testtabl (empno int, deptno int, name
```
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)
```

Access rule examples

The following 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
```
Using row-level access control

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

<table>
<thead>
<tr>
<th>empno</th>
<th>deptno</th>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>smith</td>
<td>8282</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>smith</td>
<td>9999</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>smith</td>
<td>8888</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>jones</td>
<td>9999</td>
</tr>
</tbody>
</table>
(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

<table>
<thead>
<tr>
<th>empno</th>
<th>deptno</th>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>smith</td>
<td>3245</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>smith</td>
<td>8282</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>smith</td>
<td>9999</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>smith</td>
<td>8888</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>jones</td>
<td>9999</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>jones</td>
<td>9999</td>
</tr>
</tbody>
</table>
Access rules and alter table command

When the table owner uses the `alter table` command, Adaptive Server disables access rules during the execution of the command and enables them upon completion of the command. The access rules are disabled to avoid filtering the table data during the `alter table` command.

Access rules and `bcp`

Adaptive Server enforces access rules when data is copied out of a table using the `bcp`. Adaptive Server 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.

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:

```java
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;
```
Using row-level access control

```java
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("This 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
Using row-level access control

```
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('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
```

```
go
create default clevel_def as
sec_class.GetSecVal(suser_id())
go
```

```
go
sp_bindefault clevel_def, class_level
sp_bindrule clevel, class_level
```

```
go
grant all on sec_data to public
grant all on sec_tab to public
```

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 read-only application context that Sybase provides, SYS_SESSION, to access some session-specific information. This application context is shown as Table 6-4 on page 237. You can also create your own application contexts, as described in “Creating and using application contexts” on page 233.

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.

### 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 built-in functions.

The data used in the functions is defined in a table that contains all logins for all tables, which created by the system administrator. For more information about this table, see “Using login triggers” on page 239.

- **set_appcontext()** stores:
  ```sql
  select set_appcontext ('titles', 'rlac', '1')
  ```

- **get_appcontext()** supplies two parts of a context in a session, and retrieves the third:
  ```sql
  select get_appcontext ('titles', 'rlac')
  ```
For more information on these functions and on list_appcontext and rm_appcontext, see “Creating and using application contexts” on page 233.

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 Adaptive Server 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

Functions do not have an object ID and they do not have a home database. Therefore, each database owner must grant the select privilege for the functions to the appropriate user. Adaptive Server finds the user’s default database and checks the permissions against this database. With this approach, only the owner of the users’ default database needs to grant the select privilege. If other databases should be restricted, the owner of those databases must explicitly revoke permission from the user in those databases.

Only the application context built-in functions perform data access control checks on the user when you grant and revoke privileges on them. Granting or revoking privileges for other functions has no effect in Adaptive Server.

Privileges granted to public affect only users named in the table created by the system administrator. For information about the table, see “Using login triggers” on page 239. 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
CHAPTER 6  Managing User Permissions

Creating and using application contexts

The following functions are available for creating and maintaining application contexts. For more information, see Reference Manual: Building Blocks.

- set_appcontext
- get_appcontext
- list_appcontext
- rm_appcontext

set_appcontext

Sets an application context name, attribute name, and attribute value, defined by the attributes of an application, for a specified user session.

```
set_appcontext ("context_name", "attribute_name", "attribute_value")
```

Parameters

- `context_name` – a row that specifies an application context name, saved as the datatype `char(30)`.
- `attribute_name` – a row that specifies an application context name, saved as the datatype `char(30)`
- `attribute_value` – a row that specifies an application attribute value, saved as the datatype `char(255)`.

Examples

**Example 1**  Creates an application context called CONTEXT1, with an attribute ATTR1 that has the value VALUE1:

```
select set_appcontext ("CONTEXT1", "ATTR1", "VALUE1")
---------------
0
```

**Example 2**  Shows an attempt to override the existing application context. The attempt fails, returning -1:

```
select set_appcontext ("CONTEXT1", "ATTR1", "VALUE1")
---------------
-1
```

**Example 3**  Shows how set_appcontext can include a datatype conversion in the value:

```
declare @val numeric
select @val = 20
select set_appcontext ("CONTEXT1", "ATTR2", convert(char(20), @val))
---------------
```

Using row-level access control

0

Example 4  Shows the result when a user without appropriate permissions attempts to set the application context. The attempt fails, returning -1:

```sql
select set_appcontext("CONTEXT1", "ATTR2", "VALUE1")
-------------
-1
```

Usage

- `set_appcontext` returns 0 for success and -1 for failure.
- If you set values that already exist in the current session, `set_appcontext` returns -1.
- `set_appcontext` cannot override the values of an existing application context. To assign new values to a context, remove the context and re-create it using the new values.
- `set_appcontext` saves attributes as `char` datatypes. If you create an access rule that must compare the attribute value to another datatype, the rule should convert the `char` data to the appropriate datatype.
- All arguments in this function are required.

get_appcontext

Returns the value of the attribute in a specified context.

```sql
get_appcontext ("context_name", "attribute_name")
```

Parameters

- `context_name` – a row specifying an application context name, saved as datatype `char(30)`.
- `attribute_name` – a row specifying an application context attribute name, saved as datatype `char(30)`.

Examples

Example 1  Shows VALUE1 returned for ATTR1:

```sql
select get_appcontext ("CONTEXT1", "ATTR1")
----------
VALUE1
```

Example 2  ATTR1 does not exist in CONTEXT2:

```sql
select get_appcontext ("CONTEXT2", "ATTR1")
----------
NULL
```

Example 3  Shows the result when a user without appropriate permissions attempts to get the application context:

```sql
select get_appcontext ("CONTEXT1", "ATTR2")
```
**CHAPTER 6  Managing User Permissions**

```
select permission denied on built-in get_appcontext,  
database dbid  
---------  
-1
```

**Usage**

- `get_appcontext` returns 0 for success and -1 for failure.
- If the attribute you require does not exist in the application context, `get_appcontext` returns “null.”
- `get_appcontext` saves attributes as char datatypes. If you create an access rule that compares the attribute value to other datatypes, the rule should convert the char data to the appropriate datatype.
- All arguments in this function are required.

**list_appcontext**

Lists all the attributes of all the contexts in the current session.

```
list_appcontext ("context_name")
```

**Parameters**

- `context_name` – names all the application context attributes in the session.
  `list_appcontext` has a datatype of char(30).

**Examples**

**Example 1**  Shows the results of a user with appropriate permissions listing the application contexts:

```
select list_appcontext ("*", "")
Context Name: (CONTEXT1)
Attribute Name: (ATTR1) Value: (VALUE2)
Context Name: (CONTEXT2)
Attribute Name: (ATTR1) Value: (VALUE!)
---------
0
```

**Example 2**  Shows a user without appropriate permissions attempting to list the application contexts. The attempt fails, returning -1.

```
select list_appcontext ()
Select permission denied on built-in
list_appcontext, database DBID  
---------  
-1
```

**Usage**

- `list_appcontext` returns 0 for success and -1 for failure.
- Since built-in functions do not return multiple result sets, the client application receives `list_appcontext` returns as messages.
Using row-level access control

Permissions
To use list_appcontext, the user must have appropriate permissions. For more information, see “Setting permissions for using application context functions” on page 231.

rm_appcontext

Removes a specific application context, or all application contexts.

\[ \text{rm_appcontext ("context_name", "attribute_name")} \]

Parameters
- context_name – a row specifying an application context name, saved as datatype char(30).
- attribute_name – a row specifying an application context attribute name, saved as datatype char(30).

Examples

Example 1 Uses an asterisk ("*"") to remove all attributes in the specified context.

\[
\text{select rm_appcontext ("CONTEXT1", "*"})
\]

---------
0

Example 2 Uses an asterisk ("*"") to remove all the contexts and attributes.

\[
\text{select rm_appcontext ("*", "*"})
\]

---------
0

Example 3 Shows a user attempting to remove a nonexistent context. The attempt fails, returning -1.

\[
\text{select rm_appcontext ("NON_EXISTING_CTX", "ATTR2")}
\]

---------
-1

Example 4 Shows the result of a user without appropriate permissions attempting to remove an application context.

\[
\text{select rm_appcontext ("CONTEXT1", "ATTR2")}
\]

---------
-1

Usage
- rm_appcontext returns 0 for success, -1 for failure.
- All arguments in this function are required.
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 6-4: SYS_SESSION attributes and values

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Login name</td>
</tr>
<tr>
<td>hostname</td>
<td>Host name from which the client has connected</td>
</tr>
<tr>
<td>applname</td>
<td>Name of the application as set by the client</td>
</tr>
<tr>
<td>suserid</td>
<td>User ID of the user in the current database</td>
</tr>
<tr>
<td>groupid</td>
<td>Group ID of the user in the current database</td>
</tr>
<tr>
<td>dbid</td>
<td>ID of the user’s current database</td>
</tr>
<tr>
<td>dbname</td>
<td>Current database</td>
</tr>
<tr>
<td>spid</td>
<td>Server process ID</td>
</tr>
<tr>
<td>proxy_suserid</td>
<td>The server user ID of the proxy</td>
</tr>
<tr>
<td>client_name</td>
<td>Client name set by the middle-tier application, using set client_name</td>
</tr>
<tr>
<td>client_applname</td>
<td>Client application name set by the middle-tier application, using set client_applname</td>
</tr>
<tr>
<td>client_hostname</td>
<td>Client host name set by the middle-tier application, using set client_hostname</td>
</tr>
<tr>
<td>language</td>
<td>Current language the client is using by default or after using set language (@@language)</td>
</tr>
<tr>
<td>character_set</td>
<td>Character set the client is using (@@client_csname)</td>
</tr>
<tr>
<td>dateformat</td>
<td>Date expected by the client, set using set dateformat</td>
</tr>
<tr>
<td>is_showplan_on</td>
<td>Returns YES if set showplan is on, NO if it is off</td>
</tr>
<tr>
<td>is_noexec_on</td>
<td>Returns YES if set no exec is on, NO if it is off</td>
</tr>
</tbody>
</table>

Solving a problem using an access rule and ACF

This section 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.
Using row-level access control

- 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.

```sql
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)
Using login triggers

Note Some information in this section is from the article "Login Triggers in ASE 12.5" at http://www.sypron.nl/logtrig.html. Copyright 1998–2002, Rob Verschoor/ Sypron B.V.

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.

Creating 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:

```sql
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:

```sql
create procedure loginproc as
    declare @appname varchar(20)
    declare @attr varchar(20)
    declare @value varchar(20)
    declare @retvalue int
    declare apctx cursor for
```
Using row-level access control

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 @retval = set_appcontext (rtrim (@appname),
rtrim(@attr), rtrim(@value))
  fetch apctx into @appname, @attr, @value
end

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.

Configuring 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 Adaptive Server 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 Adaptive Server 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. Adaptive Server 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.

Dropping and changing the login trigger

Once you have configured a stored procedure as a login trigger, you cannot drop it. You must unconfigure it first, either by dropping the login trigger altogether, or by changing the login trigger to a different stored procedure. To drop the login trigger, enter:
CHAPTER 6  Managing User Permissions

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"

To display the current login trigger, use sp_displaylogin:

sp_displaylogin my_login

Default Database: my_db
Default Language:
Auto Login Script: my_proc

Executing a login trigger

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, Adaptive Server 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.

Understanding 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 Adaptive Server error log file, as are some, but not all, error messages.
Using row-level access control

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 Adaptive Server error log as:

```
(....) background task message: Hello!
(....) background task error 123456: This is test message 123456
```

However, not all output goes to the Adaptive Server error log:

- No result sets from `select` statements (which are normally sent to a client connection) appear anywhere, not even in the Adaptive Server 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.

Using login triggers for other applications

Login triggers are part of the row-level access control feature in Adaptive Server. 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 Adaptive Server. However, you can use login triggers for other purposes as well.

Limiting the number of concurrent connections

The following example limits the number of concurrent connections to Adaptive Server that a specific login can make. Each of the commands described in steps 1 and 2 in the example are executed in the default database of the user for whom the access needs to be restricted:

1. As system administrator, create the `limit_user_sessions` stored procedure:

   ```sql
   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
end

grant exec on limit_user_sessions to public

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

3 Now, when user “bob” creates a third session for Adaptive Server, this session is terminated by the login trigger calling the syb_quit() function:

% isql -SASE125 -Ubob -Pbobpassword
1> select 1
2> go
CT-LIBRARY error:
cr_results(): network packet layer: internal net library error: Net-Library operation terminated due to disconnect

4 This message appears in the Adaptive Server error log file:

   (... background task message: Aborting login [ my_login]: exceeds session limit [2]

This example describes how system administrators can create a login trigger to enforce time-based restrictions on user sessions. Each of the commands described in steps 1 – 4 are executed in the default database of the user for whom the access needs to be restricted:

1 As system administrator, create this table:

    create table access_times (suid int not null,
                              dayofweek tinyint,
2 As system administrator, insert the following rows in table access_times. These rows indicate that user “bob” is allowed to log into Adaptive Server on Mondays between 9:00am and 5:00pm, and user “mark” is allowed to login to Adaptive Server on Tuesdays between 9:00am and 5:00pm.

```sql
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:

```sql
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 "Abort login [%1!]: login attempt past
    normal working hours", @loginname

    -- abort this session
    return -4
end
```

Using row-level access control
grant exec on limit_access_time to public

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” can successfully create a session:

isql -Ubob -Ppassword
1> select 1
2> go
-----------
1
(1 row affected)

However, user “mark” is denied access to Adaptive Server:

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

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.

Login trigger restrictions

The following 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 Adaptive Server.

• 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 Adaptive Server error log, do not use login triggers. Always check the Adaptive Server error log for error messages.

• For Adaptive Server 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.

  Adaptive Server versions 15.0.1, 12.5.4, and earlier required that you start Adaptive 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 Adaptive Server 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 Adaptive 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 Adaptive Server error log and the user connection closes immediately (though isql still shows a command prompt).

  Adaptive Server 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 Adaptive Server error log:

Procedure my_proc expects parameter @param1, which 
was not supplied...

Disabling 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.

Note If the login trigger returns a negative number, the login fails.

Exporting set options from a login trigger

Adaptive Server 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
Using row-level access control

- 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
Setting 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`. 
Using row-level access control
# Granting Predicated Privileges

## Introduction to 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 Adaptive Server during data access. If the condition expressed by the predicate is not met for any row of data, Adaptive Server withholds that row 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.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to predicated privileges</td>
<td>251</td>
</tr>
<tr>
<td>Commands used for predicated privileges</td>
<td>252</td>
</tr>
<tr>
<td>Configuring Adaptive Server to use predicated privileges</td>
<td>253</td>
</tr>
<tr>
<td>Granting predicated privileges</td>
<td>254</td>
</tr>
<tr>
<td>Revoking predicated privileges</td>
<td>257</td>
</tr>
<tr>
<td>How Adaptive Server saves predicated privileges in sysprotects</td>
<td>258</td>
</tr>
<tr>
<td>Predicated role activation</td>
<td>259</td>
</tr>
<tr>
<td>Combining predicates to enforce row-level privileges</td>
<td>260</td>
</tr>
<tr>
<td>Understanding SQL behavior with predicated privileges</td>
<td>264</td>
</tr>
<tr>
<td>Chain-of-ownership effect on predicated privileges</td>
<td>265</td>
</tr>
<tr>
<td>ansi_permissions and predicated privileges</td>
<td>266</td>
</tr>
<tr>
<td>Permissions on accesses made by predicates</td>
<td>268</td>
</tr>
<tr>
<td>Using triggers with predicated privileges</td>
<td>269</td>
</tr>
<tr>
<td>Recompiling predicated privileges</td>
<td>269</td>
</tr>
<tr>
<td>Disallowing recursive predicate processing</td>
<td>271</td>
</tr>
<tr>
<td>Information leakage through predicates</td>
<td>271</td>
</tr>
</tbody>
</table>
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. Adaptive Server filters the data for specific users according to the predicated privileges granted directly to the user, or granted indirectly through the user’s groups or roles.

Table 7-1 compares the advantages of predicated privileges over the access rules provided by versions of Adaptive Server earlier than 15.7, ESD #2:

<table>
<thead>
<tr>
<th><strong>Predicate scope</strong></th>
<th>Access rules</th>
<th>Predicated privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can refer only to the column to which the rule is bound. Multiple access rules are required to reference more than one column.</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Statement scope</strong></th>
<th>Access rules</th>
<th>Predicated privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>All rules apply to select, update, and delete. Does not allow different restrictions across different kinds of access.</td>
<td>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.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Combining predicates</strong></th>
<th>Access rules</th>
<th>Predicated privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>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).</td>
<td>A predicated privilege can construct an arbitrarily complex Boolean expression. You may combine multiple privileges against the same object based on well-defined rules.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Restricting the scope to a subject</strong></th>
<th>Access rules</th>
<th>Predicated privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any restriction directed at a specific subject must be expressed as part of the rule, such as where name = user.</td>
<td>You can apply a predicate to selected users, groups, or roles without the administrator needing to introduce complicated logic into the predicate.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Integration with permissions system</strong></th>
<th>Access rules</th>
<th>Predicated privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoupled from SQL authorization. Requires rule analysis to understand how users are restricted from some action.</td>
<td>Strongly enforced through Adaptive Server permission system. Diagnostic commands allow an application developer to predict how the predicates modify a query.</td>
<td></td>
</tr>
</tbody>
</table>

---

**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.

Configuring Adaptive Server to use predicated privileges

To configure Adaptive Server to use predicated privileges:

1 Enable the ASE_SECDIR license, required for Adaptive Server Security and Directory Services. See the Sybase Software Asset Management Users Guide.

2 Configure the enable predicated privileges configuration parameter. (see below)

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 Adaptive Server requires. 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, Adaptive Server creates a separate plan for each user executing the procedure. In this case, you may need to increase the size of the procedure cache.
Granting predicated privileges

enable predicated privileges

<table>
<thead>
<tr>
<th>Summary information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
</tr>
<tr>
<td>Valid values</td>
</tr>
</tbody>
</table>

enable predicated privileges enables or disables predicated privileges.

Granting 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:

```sql
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:

```sql
grant select on employee (salary)
    where ename = USER or
    emgr = USER to eng_role
```
The statement filters rows on the select command (including selects 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.

Adaptive Server 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.

**Granting 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:

```sql
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:

```sql
grant select on employee(ename, edept)
  to eng_role

grant select on employee(ephone)
    where user_id() = emgr to eng_role
```

**Granting 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.
Granting predicated privileges

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.

Granting 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'

Adaptive Server runs the statement internally as:

delete employee where status = 'terminated' and edept = 'sales'

Using predicated privileges to enforce a data privacy policy

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
Revoking 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`.

(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
How Adaptive Server saves predicated privileges in sysprotects

- 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.

```sql
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.

```sql
revoke select on t1 with no predicates
```

See the Reference Manual: Commands.

How Adaptive Server 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, Adaptive Server generally retains lower-precedence rows in sysprotects.

In this example, the second non-predicated grant to public causes Adaptive Server to remove the earlier grant to an individual user1:

```sql
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. Adaptive Server does not remove the grant to user1, even though the grantee, public, has a higher precedence:

```sql
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:

```sql
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.

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.

**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.

**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, Adaptive Server returns an error.

```sql
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:

```sql
grant role sa_role
    where
        (select get_appcontext('SYS_SESSION', 'applname'))
        = 'mon_resource'
    to loginprof_admin
```

If a role activation predicate evaluates to false, Adaptive Server returns an error for the `set role` command, or silently does not set a default or automatically activated role during login.

**Combining predicates to enforce row-level privileges**

Adaptive Server combines and attaches predicates to `update`, `delete`, or `select` commands on a specific table, 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.
CHAPTER 7  Granting Predicated Privileges

- 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.

Adaptive Server 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')
```

Example 1  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:

```sql
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, Adaptive
Server notices that columns eid and ename are affected by the two predicated
grant commands:

```sql
select eid, ename from employee
```

Adaptive Server internally executes:

```sql
select eid, ename from employee
  where edept = "sales" or
  ezip like "97%"
```

Adaptive Server then returns:

<table>
<thead>
<tr>
<th>eid</th>
<th>ename</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Bill</td>
</tr>
<tr>
<td>1009</td>
<td>Sally</td>
</tr>
<tr>
<td>1006</td>
<td>Sue</td>
</tr>
</tbody>
</table>

**Example 2**  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, Adaptive
Server 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:

```sql
grant select on employee (esalary)
  where edept = "sales"
  to sales_mgr_role

grant select on employee (eid, ename, eaddr, ezip)
  where ezip like "97%"
```
When Ben executes this statement, Adaptive Server notices that columns ename and esalary are affected by the different grant commands:

```
select ename, esalary from employee
```

Adaptive Server internally executes:

```
select ename, esalary from employee
where edept = "sales" and ezip like "97%"
```

Adaptive Server then returns:

```
ename   esalary
sally   50000
```

**Example 3**  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
```

Adaptive Server 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)

Adaptive Server internally executes:
update employees
set esalary = esalary + (esalary * 0.05)
where ezip like "97%"
and esalary < $100000

Adaptive Server then updates the row as:
edid   ename    esalary  
1009    Sally    $52500

Understanding SQL behavior with 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 display 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:

```sql
sp_helptext pred1
# Lines of Text
-------------------
1

text
-------------------
---
grant select on tab1 where coll = 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.


## 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, Adaptive Server does not apply the predicates 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:

```sql
grant select on employee
   where dept = 'payroll'
to priscilla
```

When Priscilla enters:

```sql
select name, phone from employee
   where city = 'SFO'
```

Adaptive Server returns the names and phone numbers 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, Adaptive Server detects the ownership chain between employee and employee_addresses:

```sql
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.

### 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, Adaptive Server checks for select permission privileges on these columns only if ansi_permissions is enabled for the session. Adaptive Server 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:

```sql
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, Adaptive Server 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, Adaptive Server 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.
Permissions on accesses made by predicates

- Enabled – the where clause on a predicated grant for select access qualifies the row set that may be updated or deleted.

**Note** Either one or both of the predicates from the grant commands in the examples above are applied.

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, Adaptive Server 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.

Adaptive Server 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).

```sql
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, Adaptive Server performs these access checks:

- Alice’s direct or indirect select permission on purchases
- Bob’s direct or indirect permission to select from the privacy_db..choices table

**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.

---

**Using triggers with predicated privileges**

Adaptive Server 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.

**Recompiling predicated privileges**

If a stored procedure, trigger, or SQL function accesses tables that are controlled by predicated privileges, Adaptive Server 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.
Recompiling predicated privileges

To prevent constant recompiling for different users executing the same procedure, Adaptive Server 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, Adaptive Server chooses a copy of the plan that is less likely to require compiling.

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.

the user ID for whom the plan was compiled, and the IDs of roles and groups that authorized the various accesses made by the procedure.

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. Adaptive Server 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.
Disallowing recursive predicate processing

Adaptive Server 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, Adaptive Server ignores any default roles that have been granted with an activation predicate.

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 Adaptive Server 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:

```sql
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 = 'IO'
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.
CHAPTER 8  
Using Granular Permissions

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to granular permissions</td>
<td>273</td>
</tr>
<tr>
<td>Configuring Adaptive Server to use granular permissions</td>
<td>274</td>
</tr>
<tr>
<td>System privileges</td>
<td>274</td>
</tr>
<tr>
<td>Effect of privileges as part of system-defined roles</td>
<td>275</td>
</tr>
<tr>
<td>Permission management</td>
<td>276</td>
</tr>
<tr>
<td>Privileges granted to system-defined roles</td>
<td>284</td>
</tr>
<tr>
<td>Privileges assigned to the database owner</td>
<td>288</td>
</tr>
<tr>
<td>Roles added with granular permissions</td>
<td>290</td>
</tr>
<tr>
<td>Default roles granted to the system administrator</td>
<td>292</td>
</tr>
<tr>
<td>Limiting the power of the system administrator and database owner</td>
<td>293</td>
</tr>
<tr>
<td>Enable granular permissions and sybsecurity</td>
<td>294</td>
</tr>
<tr>
<td>Logging in to a locked-out Adaptive Server</td>
<td>295</td>
</tr>
<tr>
<td>General use scenarios</td>
<td>296</td>
</tr>
<tr>
<td>System table master.dbo.sysprotects</td>
<td>298</td>
</tr>
<tr>
<td>Database user use db._user</td>
<td>299</td>
</tr>
<tr>
<td>Grantable system privileges</td>
<td>300</td>
</tr>
</tbody>
</table>

Introduction to 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 regranted them to the roles.

Configuring Adaptive Server to use granular permissions

To configure Adaptive Server to use granular permissions:

1. Enable the security and directory services license, ASE_SECDIRS. See the Sybase Software Asset Management Users Guide.

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.

To turn on enable granular permissions, Adaptive Server 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, Adaptive Server verifies the server has at least one unlocked account with the sso_role, and that at least one unlocked account has the sa_role.

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. Adaptive Server stores the permission information for server-wide privileges in master..dbo.sysprotects. See Table 8-15 on page 301 for a list of server-wide privileges.
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. Adaptive Server stores the permission information for database-wide privileges in `database_name.dbo.sysprotects`. See Table 8-16 on page 315 for a list of database-wide privileges.

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:

```sql
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`:

```sql
grant create any object to joe
```

**Effect of privileges as part of system-defined roles**

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 Adaptive Server 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.
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.

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. See Table 8-1 and Table 8-2 respectively, for a list of these privileges.
Table 8-1: 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 8-2: 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, Adaptive Server ensures the server contains at least one unlocked user account with manage security permissions privilege.
manage server permissions privilege

Users with manage server permissions privilege can grant and revoke server-wide privilege. See Table 8-3 for a list of these privileges.
### Table 8-3: system privileges managed by manage server permissions

<table>
<thead>
<tr>
<th>privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow exceptional login</td>
</tr>
<tr>
<td>checkpoint any database</td>
</tr>
<tr>
<td>checkpoint (on any database except sybsecurity)</td>
</tr>
<tr>
<td>connect</td>
</tr>
<tr>
<td>create database</td>
</tr>
<tr>
<td>dbcc checkallocate any database</td>
</tr>
<tr>
<td>dbcc checkcatalog any database</td>
</tr>
<tr>
<td>dbcc checkdb any database</td>
</tr>
<tr>
<td>dbcc checkindex any database</td>
</tr>
<tr>
<td>dbcc checkstorage any database</td>
</tr>
<tr>
<td>dbcc checktable any database</td>
</tr>
<tr>
<td>dbcc checkverify any database</td>
</tr>
<tr>
<td>dbcc fix_text any database</td>
</tr>
<tr>
<td>dbcc indexalloc any database</td>
</tr>
<tr>
<td>dbcc reindex any database</td>
</tr>
<tr>
<td>dbcc tablealloc any database</td>
</tr>
<tr>
<td>dbcc textalloc any database</td>
</tr>
<tr>
<td>dbcc tune</td>
</tr>
<tr>
<td>dump any database</td>
</tr>
<tr>
<td>dump database (on any database except sybsecurity)</td>
</tr>
<tr>
<td>kill</td>
</tr>
<tr>
<td>kill any process</td>
</tr>
<tr>
<td>load any database</td>
</tr>
<tr>
<td>load database (on any database except sybsecurity)</td>
</tr>
<tr>
<td>manage any database</td>
</tr>
<tr>
<td>manage any ESP</td>
</tr>
<tr>
<td>manage any thread pool</td>
</tr>
<tr>
<td>manage cluster</td>
</tr>
<tr>
<td>manage data cache</td>
</tr>
<tr>
<td>manage disk</td>
</tr>
<tr>
<td>manage dump configurations</td>
</tr>
<tr>
<td>manage lock promotion threshold</td>
</tr>
<tr>
<td>manage resource limit</td>
</tr>
<tr>
<td>manage server</td>
</tr>
<tr>
<td>manage server configuration</td>
</tr>
<tr>
<td>manage server permissions</td>
</tr>
</tbody>
</table>
**Permission management**

<table>
<thead>
<tr>
<th>manage execution classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>map external file</td>
</tr>
<tr>
<td>monitor server replication</td>
</tr>
<tr>
<td>mount any database</td>
</tr>
<tr>
<td>online any database</td>
</tr>
<tr>
<td>online database (on any database except syssecurity)</td>
</tr>
<tr>
<td>own any database</td>
</tr>
<tr>
<td>own database (on any database except syssecurity)</td>
</tr>
<tr>
<td>quiesce any database</td>
</tr>
<tr>
<td>set switch</td>
</tr>
<tr>
<td>set tracing</td>
</tr>
<tr>
<td>set tracing any process</td>
</tr>
<tr>
<td>show switch</td>
</tr>
<tr>
<td>shutdown</td>
</tr>
<tr>
<td>unmount any database</td>
</tr>
<tr>
<td>use any database</td>
</tr>
<tr>
<td>use database (on any database except syssecurity)</td>
</tr>
</tbody>
</table>

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, Adaptive Server ensures the server contains at least one unlocked user account with manage server permissions privileges.

**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. See Table 8-4 for a list of these privileges.

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.
Permission management

Table 8-4: Privileges managed by manage database permissions privilege

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter any object owner</td>
</tr>
<tr>
<td>alter any table</td>
</tr>
<tr>
<td>create any default</td>
</tr>
<tr>
<td>create any function</td>
</tr>
<tr>
<td>create any index</td>
</tr>
<tr>
<td>create any object</td>
</tr>
<tr>
<td>create any procedure</td>
</tr>
<tr>
<td>create any rule</td>
</tr>
<tr>
<td>create any table</td>
</tr>
<tr>
<td>create any trigger</td>
</tr>
<tr>
<td>create any view</td>
</tr>
<tr>
<td>create default</td>
</tr>
<tr>
<td>create function</td>
</tr>
<tr>
<td>create procedure</td>
</tr>
<tr>
<td>create rule</td>
</tr>
<tr>
<td>create table</td>
</tr>
<tr>
<td>create trigger</td>
</tr>
<tr>
<td>create view</td>
</tr>
<tr>
<td>dbcc checkalloc</td>
</tr>
<tr>
<td>dbcc checkcatalog</td>
</tr>
<tr>
<td>dbcc checkdb</td>
</tr>
<tr>
<td>dbcc checkindex</td>
</tr>
<tr>
<td>dbcc checkstorage</td>
</tr>
<tr>
<td>dbcc checktable</td>
</tr>
<tr>
<td>dbcc checkverify</td>
</tr>
<tr>
<td>dbcc fix_text</td>
</tr>
<tr>
<td>dbcc indexalloc</td>
</tr>
<tr>
<td>dbcc reindex</td>
</tr>
<tr>
<td>dbcc tablealloc</td>
</tr>
<tr>
<td>dbcc textalloc</td>
</tr>
<tr>
<td>delete any table</td>
</tr>
<tr>
<td>drop any default</td>
</tr>
<tr>
<td>drop any function</td>
</tr>
<tr>
<td>drop any object</td>
</tr>
<tr>
<td>drop any procedure</td>
</tr>
<tr>
<td>drop any rule</td>
</tr>
</tbody>
</table>
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. See Table 8-5 for a list of object privileges.
Privileges granted to system-defined roles

Table 8-5: Object permissions managed by manage any object permission privilege

<table>
<thead>
<tr>
<th>Permission</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete</td>
</tr>
<tr>
<td>delete statistics</td>
</tr>
<tr>
<td>execute</td>
</tr>
<tr>
<td>identity_insert</td>
</tr>
<tr>
<td>identity_update</td>
</tr>
<tr>
<td>insert</td>
</tr>
<tr>
<td>references</td>
</tr>
<tr>
<td>select</td>
</tr>
<tr>
<td>transfer table</td>
</tr>
<tr>
<td>truncate table</td>
</tr>
<tr>
<td>update</td>
</tr>
<tr>
<td>update statistics</td>
</tr>
</tbody>
</table>

Privileges granted to system-defined roles

Default privileges granted to system defined roles are listed in Table 8-6 through Table 8-10.
### Table 8-6: Privileges granted to sa_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow exceptional login</td>
</tr>
<tr>
<td>checkpoint any database</td>
</tr>
<tr>
<td>connect</td>
</tr>
<tr>
<td>create database</td>
</tr>
<tr>
<td>dbcc checkalloc any database</td>
</tr>
<tr>
<td>dbcc checkcatalog any database</td>
</tr>
<tr>
<td>dbcc checkdb any database</td>
</tr>
<tr>
<td>dbcc checkindex any database</td>
</tr>
<tr>
<td>dbcc checkstorage any database</td>
</tr>
<tr>
<td>dbcc checktable any database</td>
</tr>
<tr>
<td>dbcc checkverify any database</td>
</tr>
<tr>
<td>dbcc fix_text any database</td>
</tr>
<tr>
<td>dbcc indexalloc any database</td>
</tr>
<tr>
<td>dbcc reindex any database</td>
</tr>
<tr>
<td>dbcc tablealloc any database</td>
</tr>
<tr>
<td>dbcc textalloc any database</td>
</tr>
<tr>
<td>dbcc tune</td>
</tr>
<tr>
<td>dump any database</td>
</tr>
<tr>
<td>kill any process</td>
</tr>
<tr>
<td>load any database</td>
</tr>
<tr>
<td>manage any database</td>
</tr>
<tr>
<td>manage any ESP</td>
</tr>
<tr>
<td>manage any execution class</td>
</tr>
<tr>
<td>manage any thread pool</td>
</tr>
<tr>
<td>manage cluster</td>
</tr>
<tr>
<td>manage data cache</td>
</tr>
<tr>
<td>manage disk</td>
</tr>
<tr>
<td>manage dump configuration</td>
</tr>
<tr>
<td>manage lock promotion threshold</td>
</tr>
<tr>
<td>manage resource limit</td>
</tr>
<tr>
<td>manage server</td>
</tr>
<tr>
<td>manage server configuration</td>
</tr>
<tr>
<td>manage server permissions</td>
</tr>
<tr>
<td>map external file</td>
</tr>
<tr>
<td>mount any database</td>
</tr>
<tr>
<td>online any database</td>
</tr>
<tr>
<td>own any database</td>
</tr>
</tbody>
</table>
Privileges granted to system-defined roles

- 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 8-7: Privileges granted to sso_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter any object owner (in any database)</td>
</tr>
<tr>
<td>change password</td>
</tr>
<tr>
<td>decrypt any table (in any database)</td>
</tr>
<tr>
<td>manage any encryption key (in any database)</td>
</tr>
<tr>
<td>manage any login</td>
</tr>
<tr>
<td>manage any login profile</td>
</tr>
<tr>
<td>manage any remote login</td>
</tr>
<tr>
<td>manage any user (in any database)</td>
</tr>
<tr>
<td>manage auditing</td>
</tr>
<tr>
<td>manage roles</td>
</tr>
<tr>
<td>manage security configuration</td>
</tr>
<tr>
<td>manage security permissions</td>
</tr>
<tr>
<td>select on authmech</td>
</tr>
<tr>
<td>show switch</td>
</tr>
<tr>
<td>set tracing any process</td>
</tr>
<tr>
<td>update any security catalog (in any database)</td>
</tr>
</tbody>
</table>

Table 8-8: Privileges granted to oper_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkpoint any database</td>
</tr>
<tr>
<td>dump any database</td>
</tr>
<tr>
<td>load any database</td>
</tr>
<tr>
<td>manage dump configuration</td>
</tr>
<tr>
<td>online any database</td>
</tr>
<tr>
<td>use any database</td>
</tr>
</tbody>
</table>

Table 8-9: Privileges granted to replication_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkpoint any database</td>
</tr>
<tr>
<td>dump any database</td>
</tr>
<tr>
<td>load any database</td>
</tr>
<tr>
<td>manage replication (in any database)</td>
</tr>
<tr>
<td>monitor server replication</td>
</tr>
<tr>
<td>online any database</td>
</tr>
<tr>
<td>quiesce any database</td>
</tr>
<tr>
<td>truncate any table (in any database)</td>
</tr>
<tr>
<td>truncate any audit table (in sybsecurity)</td>
</tr>
</tbody>
</table>

Table 8-10: Privileges granted to keycustodian_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage any encryption key</td>
</tr>
</tbody>
</table>
Privileges assigned to the database owner

Use `sp_restore_system_role` (which requires manage security permissions privileges) to restore a role or database owner to the default role privilege configuration.

**Privileges assigned to the database owner**

Default privileges granted to the database owner are listed in Table 8-11.
Table 8-11: Privileges granted to database owners by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter any object owner</td>
</tr>
<tr>
<td>create default</td>
</tr>
<tr>
<td>create function</td>
</tr>
<tr>
<td>create procedure</td>
</tr>
<tr>
<td>create rule</td>
</tr>
<tr>
<td>create table</td>
</tr>
<tr>
<td>create trigger</td>
</tr>
<tr>
<td>create view</td>
</tr>
<tr>
<td>dbcc checkalloc</td>
</tr>
<tr>
<td>dbcc checkcatalog</td>
</tr>
<tr>
<td>dbcc checkedb</td>
</tr>
<tr>
<td>dbcc checkindex</td>
</tr>
<tr>
<td>dbcc checkstorage</td>
</tr>
<tr>
<td>dbcc checktable</td>
</tr>
<tr>
<td>dbcc checkverify</td>
</tr>
<tr>
<td>dbcc fix_text</td>
</tr>
<tr>
<td>dbcc indexalloc</td>
</tr>
<tr>
<td>dbcc reindex</td>
</tr>
<tr>
<td>dbcc tablealloc</td>
</tr>
<tr>
<td>dbcc textalloc</td>
</tr>
<tr>
<td>manage abstract plans</td>
</tr>
<tr>
<td>manage any user</td>
</tr>
<tr>
<td>manage checkstorage</td>
</tr>
<tr>
<td>manage database</td>
</tr>
<tr>
<td>manage database permissions</td>
</tr>
<tr>
<td>manage database permissions</td>
</tr>
<tr>
<td>manage replication</td>
</tr>
<tr>
<td>report checkstorage</td>
</tr>
<tr>
<td>select any audit table</td>
</tr>
<tr>
<td>setuser</td>
</tr>
<tr>
<td>truncate any audit table</td>
</tr>
</tbody>
</table>

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.

Roles added with granular permissions

Granular permissions adds the role sa_serverprivs_role.

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 Adaptive Server 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.

Note Sybase 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.

Default privileges granted to sa_serverprivs_role are listed in Table 8-13.
### Table 8-12: Privileges granted to sa_serverprivs_role by default

<table>
<thead>
<tr>
<th>Privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow exceptional login</td>
</tr>
<tr>
<td>change password</td>
</tr>
<tr>
<td>checkpoint any database</td>
</tr>
<tr>
<td>connect</td>
</tr>
<tr>
<td>create database</td>
</tr>
<tr>
<td>dbcc checkalloc any database</td>
</tr>
<tr>
<td>dbcc checkcatalog any database</td>
</tr>
<tr>
<td>dbcc checkdb any database</td>
</tr>
<tr>
<td>dbcc checkindex any database</td>
</tr>
<tr>
<td>dbcc checkstorage any database</td>
</tr>
<tr>
<td>dbcc chocketable any database</td>
</tr>
<tr>
<td>dbcc checkverify any database</td>
</tr>
<tr>
<td>dbcc fix_text any database</td>
</tr>
<tr>
<td>dbcc indexalloc any database</td>
</tr>
<tr>
<td>dbcc reindex any database</td>
</tr>
<tr>
<td>dbcc tablealloc any database</td>
</tr>
<tr>
<td>dbcc textalloc any database</td>
</tr>
<tr>
<td>dbcc tune</td>
</tr>
<tr>
<td>dump any database</td>
</tr>
<tr>
<td>kill any process</td>
</tr>
<tr>
<td>load any database</td>
</tr>
<tr>
<td>manage any database</td>
</tr>
<tr>
<td>manage any ESP</td>
</tr>
<tr>
<td>manage any execution class</td>
</tr>
<tr>
<td>manage any login</td>
</tr>
<tr>
<td>manage any login profile</td>
</tr>
<tr>
<td>manage any remote login</td>
</tr>
<tr>
<td>manage auditing</td>
</tr>
<tr>
<td>manage cluster</td>
</tr>
<tr>
<td>manage data cache</td>
</tr>
<tr>
<td>manage disk</td>
</tr>
<tr>
<td>manage dump configuration</td>
</tr>
<tr>
<td>manage lock promotion threshold</td>
</tr>
<tr>
<td>manage resource limit</td>
</tr>
<tr>
<td>manage roles</td>
</tr>
<tr>
<td>manage security configuration</td>
</tr>
<tr>
<td>manage security permissions</td>
</tr>
</tbody>
</table>
Default roles granted to the system administrator

By default, the login sa is granted these roles:

- 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
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:

```sql
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:

```sql
use db1
revoke setuser from dbo
```
Enable granular permissions and sybsecurity

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
```

Enable granular permissions and sybsecurity

The following 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.
CHAPTER 8  Using Granular Permissions

Logging in to a locked-out Adaptive Server

If enable granular permissions is enabled, Adaptive 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 Adaptive Server.

- `-p login_name` – specifies the login name when starting Adaptive Server so this account's password may be reset. Adaptive Server generates a random password, displays it, encrypts it, and saves it 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
```
General use scenarios

Scenario 1: Permissions for an application server user

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:

```sql
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

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:

```sql
use db2
grant manage any object permission to joe
grant manage any user to joe
```
Scenario 3: Permissions for a database backup manager

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

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

A user with the manage auditing privilege issues the following to allow information security department employee Jane to generate daily reports from the Adaptive Server auditing subsystem:

```
use sybsecurity
sp_adduser jane
grant select any audit table to jane
```
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 8-14: master.dbo.spt_values**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Permission name string.</td>
</tr>
<tr>
<td>number</td>
<td>Internal token number of the permission (sysprotects.action).</td>
</tr>
</tbody>
</table>
| 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       | Adaptive Server 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  
|           | 97 – manage any object permission |
Database user **usedb_user**

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, and 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.
Grantable system privileges

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

Adaptive Server records the grantor of manage any encryption key in sysprotects as the user ID of usedb_user.

Grantable system privileges

Table 8-15 and Table 8-16 list all grantable system 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 Table 1-22 in the grant command in Reference Manual: Commands.

**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 Table 1-22 in the grant command in Reference Manual: Commands for a complete list of privileges pairs that have an implied relationship.
### Table 8-15: Server-wide privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Privileges management</strong></td>
<td></td>
</tr>
<tr>
<td>manage security permissions</td>
<td>Granting and revoking security privileges</td>
</tr>
<tr>
<td>manage server permissions</td>
<td>Granting and revoking nonsecurity server-wide privileges not related to security.</td>
</tr>
<tr>
<td><strong>Audit management</strong></td>
<td></td>
</tr>
<tr>
<td>manage auditing</td>
<td>Acting as the database owner of <code>sybsecurity</code></td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• create database <code>sybsecurity</code></td>
</tr>
<tr>
<td></td>
<td>• alter database <code>sybsecurity</code></td>
</tr>
<tr>
<td></td>
<td>• drop database <code>sybsecurity</code></td>
</tr>
<tr>
<td></td>
<td>• set switch (7601)</td>
</tr>
<tr>
<td></td>
<td>• truncate table (audit table)</td>
</tr>
<tr>
<td></td>
<td>Executing these system store procedures:</td>
</tr>
<tr>
<td></td>
<td>• <code>sp_addaudittable</code></td>
</tr>
<tr>
<td></td>
<td>• <code>sp_audit</code></td>
</tr>
<tr>
<td></td>
<td>• <code>sp_displayaudit</code></td>
</tr>
<tr>
<td><strong>Login and role management</strong></td>
<td></td>
</tr>
<tr>
<td>allow exceptional login</td>
<td>Granting login to the server when:</td>
</tr>
<tr>
<td></td>
<td>• Server maximum connection limit exceeded</td>
</tr>
<tr>
<td></td>
<td>• Master database is in restore mode</td>
</tr>
<tr>
<td></td>
<td>• Server is in shutdown</td>
</tr>
<tr>
<td></td>
<td>• Recovery is in progress (during server restart)</td>
</tr>
<tr>
<td>change password</td>
<td>Executing this command:</td>
</tr>
<tr>
<td></td>
<td>• alter login ... change password</td>
</tr>
<tr>
<td></td>
<td>Executing these system store procedures:</td>
</tr>
<tr>
<td></td>
<td>• <code>sp_password</code></td>
</tr>
<tr>
<td></td>
<td>• <code>sp_locklogin</code> (unlock any login account which was locked because the user exceeded the limit of maximum failed logins)</td>
</tr>
</tbody>
</table>
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage any login</td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• create login</td>
</tr>
<tr>
<td></td>
<td>• alter login</td>
</tr>
<tr>
<td></td>
<td>• drop login</td>
</tr>
<tr>
<td></td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addlogin</td>
</tr>
<tr>
<td></td>
<td>• sp_autoconnect</td>
</tr>
<tr>
<td></td>
<td>• sp_defaultdb</td>
</tr>
<tr>
<td></td>
<td>• sp_displaylogin</td>
</tr>
<tr>
<td></td>
<td>• sp_droplogin</td>
</tr>
<tr>
<td></td>
<td>• sp_grantlogin (Windows only)</td>
</tr>
<tr>
<td></td>
<td>• sp_helpmaplogin</td>
</tr>
<tr>
<td></td>
<td>• sp_logininfo (Windows only)</td>
</tr>
<tr>
<td></td>
<td>• sp_locklogin</td>
</tr>
<tr>
<td></td>
<td>• sp_maplogin</td>
</tr>
<tr>
<td></td>
<td>• sp_modifylogin</td>
</tr>
<tr>
<td></td>
<td>• sp_revokelogin (Windows only)</td>
</tr>
<tr>
<td></td>
<td>Executing this function:</td>
</tr>
<tr>
<td></td>
<td>• valid_user</td>
</tr>
<tr>
<td>manage any login profile</td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• create login profile</td>
</tr>
<tr>
<td></td>
<td>• alter login profile</td>
</tr>
<tr>
<td></td>
<td>• drop login profile</td>
</tr>
<tr>
<td></td>
<td>Executing this system procedure:</td>
</tr>
<tr>
<td></td>
<td>• sp_securityprofile</td>
</tr>
<tr>
<td>manage any remote login</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addexternlogin</td>
</tr>
<tr>
<td></td>
<td>• sp_addrmelogin</td>
</tr>
<tr>
<td></td>
<td>• sp_dropexternlogin</td>
</tr>
<tr>
<td></td>
<td>• sp_dropremotelogin</td>
</tr>
<tr>
<td></td>
<td>• sp_dropserver</td>
</tr>
<tr>
<td></td>
<td>• sp_remoteoption</td>
</tr>
</tbody>
</table>
### 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  
• sp_role

#### Database Management

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkpoint</td>
<td>Executing the checkpoint command for a specified database</td>
</tr>
<tr>
<td>checkpoint any database</td>
<td>Executing the checkpoint command for any database</td>
</tr>
<tr>
<td>create database*</td>
<td>Executing the create database command</td>
</tr>
</tbody>
</table>
| 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 |
**Grantable system privileges**

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
</table>
| 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_modithreshold  
|                    | • sp_placeobject  
|                    | • sp_procmode  
|                    | • sp_rename  
|                    | • sp_rebuild_text  
|                    | • sp_spaceusage (for some parameters)  

CHAPTER 8  Using Granular Permissions

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage any database (continued)</td>
<td>Executing these dbcc commands:</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(remap)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(newthreshold)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(findstranded)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(fixlogfreespace)</td>
</tr>
<tr>
<td></td>
<td>• dbr_remap</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(ltmignor)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(updusg_anchors)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(auinit)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(dmap_unlock)</td>
</tr>
<tr>
<td></td>
<td>• dbcc rebuild_text</td>
</tr>
<tr>
<td></td>
<td>• dbcc refreshids (placeobject)</td>
</tr>
<tr>
<td></td>
<td>• dbcc refreshpdes</td>
</tr>
<tr>
<td></td>
<td>• dbcc update_tmode</td>
</tr>
<tr>
<td></td>
<td>• dbcc upgrade_obj</td>
</tr>
<tr>
<td></td>
<td>Executing these functions:</td>
</tr>
<tr>
<td></td>
<td>• derived_stat</td>
</tr>
<tr>
<td></td>
<td>• identity_burn_max</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>mount any database</td>
<td>Executing the mount database command for any database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>online any database</td>
<td>Executing the online database command for any database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>online database</td>
<td>Executing the online database command for a specified database</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>own any database</td>
<td>Acting as database owner for the specified database</td>
</tr>
<tr>
<td></td>
<td>See own database for a list of operations the privilege is authorized to perform.</td>
</tr>
</tbody>
</table>
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
</table>
| 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  

Adaptive Server Enterprise

306
<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>own database (continued)</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addmessage</td>
</tr>
<tr>
<td></td>
<td>• sp_altermessage</td>
</tr>
<tr>
<td></td>
<td>• sp_dboption</td>
</tr>
<tr>
<td></td>
<td>• sp_dbremap</td>
</tr>
<tr>
<td></td>
<td>• sp_dropmessage</td>
</tr>
<tr>
<td></td>
<td>• sp_fixindex</td>
</tr>
<tr>
<td></td>
<td>• sp_forceonline_db</td>
</tr>
<tr>
<td></td>
<td>• sp_forceonline_page</td>
</tr>
<tr>
<td></td>
<td>• sp_helptext</td>
</tr>
<tr>
<td></td>
<td>• sp_logdevice</td>
</tr>
<tr>
<td></td>
<td>• sp_post_xupload</td>
</tr>
<tr>
<td></td>
<td>• sp_renamedb</td>
</tr>
<tr>
<td></td>
<td>• sp_setsuspect_granularity</td>
</tr>
<tr>
<td></td>
<td>• sp_tempdb_markdrop</td>
</tr>
<tr>
<td></td>
<td>• sp_version</td>
</tr>
<tr>
<td></td>
<td>• xp_cmdshell</td>
</tr>
<tr>
<td></td>
<td>• xp_deletemail</td>
</tr>
<tr>
<td></td>
<td>• xp_enumgroups</td>
</tr>
<tr>
<td></td>
<td>• xp_findnextmsg</td>
</tr>
<tr>
<td></td>
<td>• xp_logevent</td>
</tr>
<tr>
<td></td>
<td>• xp_readmail</td>
</tr>
<tr>
<td></td>
<td>• xp_sendmail</td>
</tr>
<tr>
<td></td>
<td>• xp_startmail</td>
</tr>
<tr>
<td></td>
<td>• xp_stopmail</td>
</tr>
<tr>
<td></td>
<td>Executing these dbcc commands:</td>
</tr>
<tr>
<td></td>
<td>• dbcc addtempdb</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair</td>
</tr>
<tr>
<td></td>
<td>• dbcc reindex</td>
</tr>
<tr>
<td>quiesce any database</td>
<td>Executing the quiesce database command for any database</td>
</tr>
<tr>
<td>unmount any database</td>
<td>Executing the unmount database command for any database</td>
</tr>
</tbody>
</table>

*Server Maintenance*
<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage any thread pool</td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• create thread pool</td>
</tr>
<tr>
<td></td>
<td>• alter thread pool</td>
</tr>
<tr>
<td></td>
<td>• drop thread pool</td>
</tr>
<tr>
<td>manage cluster</td>
<td>Managing cluster-related configurations and operations</td>
</tr>
<tr>
<td></td>
<td>Executing the shutdown cluster command (requires shutdown privilege)</td>
</tr>
<tr>
<td></td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addserver</td>
</tr>
<tr>
<td></td>
<td>• sp_cluster</td>
</tr>
<tr>
<td></td>
<td>• sp_clusterlockusage</td>
</tr>
<tr>
<td></td>
<td>• sp_dropserver</td>
</tr>
<tr>
<td></td>
<td>• sp_serveroption</td>
</tr>
<tr>
<td></td>
<td>• sp_tempdb_markdrop</td>
</tr>
<tr>
<td></td>
<td>Executing these dbcc commands:</td>
</tr>
<tr>
<td></td>
<td>• dbcc quorum</td>
</tr>
<tr>
<td></td>
<td>• dbcc set_scope_in_cluster</td>
</tr>
<tr>
<td>manage disk</td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• disk init</td>
</tr>
<tr>
<td></td>
<td>• disk refit</td>
</tr>
<tr>
<td></td>
<td>• disk reinit</td>
</tr>
<tr>
<td></td>
<td>• disk mirror</td>
</tr>
<tr>
<td></td>
<td>• disk unmirror</td>
</tr>
<tr>
<td></td>
<td>• disk remirror</td>
</tr>
<tr>
<td></td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addumpdevice</td>
</tr>
<tr>
<td></td>
<td>• sp_diskdefault</td>
</tr>
<tr>
<td></td>
<td>• sp_deviceattr</td>
</tr>
<tr>
<td></td>
<td>• sp_dropdevice</td>
</tr>
<tr>
<td></td>
<td>• sp_refit_admin</td>
</tr>
<tr>
<td></td>
<td>• sp_dbextend (required by some options)</td>
</tr>
</tbody>
</table>
CHAPTER 8 Using Granular Permissions

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage security configuration</td>
<td>Enable or disable security related configurations Executing these system procedures: • <code>sp_configure</code> (to set security-related configuration options) • <code>sp_encryption</code> • <code>sp_logintrigger</code> • <code>sp_idapadmin</code> • <code>sp_passwordpolicy</code> • <code>sp_ssladmin</code></td>
</tr>
</tbody>
</table>
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage server</td>
<td>Manage server maintenance operations</td>
</tr>
<tr>
<td></td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addlanguage</td>
</tr>
<tr>
<td></td>
<td>• sp_addserver (current security system officer)</td>
</tr>
<tr>
<td></td>
<td>• sp_clearstats</td>
</tr>
<tr>
<td></td>
<td>• sp_countmetadata</td>
</tr>
<tr>
<td></td>
<td>• sp_dbrecovery_order</td>
</tr>
<tr>
<td></td>
<td>• sp_displaylogin</td>
</tr>
<tr>
<td></td>
<td>• sp_displayroles</td>
</tr>
<tr>
<td></td>
<td>• sp_droplanguage</td>
</tr>
<tr>
<td></td>
<td>• sp_dropserver (current security system officer)</td>
</tr>
<tr>
<td></td>
<td>• sp_engine</td>
</tr>
<tr>
<td></td>
<td>• sp_errorlog</td>
</tr>
<tr>
<td></td>
<td>• sp_extengine</td>
</tr>
<tr>
<td></td>
<td>• sp_helpapptrace</td>
</tr>
<tr>
<td></td>
<td>• sp_metrics</td>
</tr>
<tr>
<td></td>
<td>• sp_monitorconfig</td>
</tr>
<tr>
<td></td>
<td>• sp_object_stats</td>
</tr>
<tr>
<td></td>
<td>• sp_processmail (Windows only)</td>
</tr>
<tr>
<td></td>
<td>• sp_reportstats</td>
</tr>
<tr>
<td></td>
<td>• sp_serveroption</td>
</tr>
<tr>
<td></td>
<td>• sp_setlangalrias</td>
</tr>
<tr>
<td></td>
<td>• sp_tempdb</td>
</tr>
<tr>
<td></td>
<td>Executing these dbcc commands:</td>
</tr>
<tr>
<td></td>
<td>• dbcc complete_xact</td>
</tr>
<tr>
<td></td>
<td>• dbcc engine</td>
</tr>
<tr>
<td></td>
<td>• dbcc forget_xact</td>
</tr>
<tr>
<td></td>
<td>• dbcc traceflags</td>
</tr>
<tr>
<td></td>
<td>Executing these functions:</td>
</tr>
<tr>
<td></td>
<td>• pssinfo</td>
</tr>
<tr>
<td></td>
<td>• valid_user</td>
</tr>
</tbody>
</table>
### 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_reconfig  
- 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 index alloc any database | Executing dbcc indexalloc in any database  
| dbcc reindex any database | Executing dbcc reindex in any database  
| dbcc table alloc any database | Executing dbcc tablealloc in any database  
| dbcc text alloc any database | Executing dbcc textalloc in any database  
| dbcc tune | Executing dbcc tune
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application Management</strong></td>
<td></td>
</tr>
<tr>
<td>manage any execution class</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>- sp_addengine</td>
</tr>
<tr>
<td></td>
<td>- sp_addexeclass</td>
</tr>
<tr>
<td></td>
<td>- sp_bindexeclass</td>
</tr>
<tr>
<td></td>
<td>- sp_clearpsexe</td>
</tr>
<tr>
<td></td>
<td>- sp_dropengine</td>
</tr>
<tr>
<td></td>
<td>- sp_dropexeclass</td>
</tr>
<tr>
<td></td>
<td>- sp_setpsexe</td>
</tr>
<tr>
<td></td>
<td>- sp_unbindexeclass</td>
</tr>
<tr>
<td>manage any ESP</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>- sp_addextendedproc</td>
</tr>
<tr>
<td></td>
<td>- sp_dropextendedproc</td>
</tr>
<tr>
<td></td>
<td>- sp_freedll</td>
</tr>
<tr>
<td></td>
<td>- sp_helpextendedproc</td>
</tr>
<tr>
<td>manage data cache</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>- sp_bindcache</td>
</tr>
<tr>
<td></td>
<td>- sp_cacheconfig</td>
</tr>
<tr>
<td></td>
<td>- sp_cachestrategy</td>
</tr>
<tr>
<td></td>
<td>- sp_logiosize</td>
</tr>
<tr>
<td></td>
<td>- sp_poolconfig</td>
</tr>
<tr>
<td></td>
<td>- sp_unbindcache</td>
</tr>
<tr>
<td></td>
<td>- sp_unbindcache_all</td>
</tr>
<tr>
<td>manage dump configuration</td>
<td>Managing dump configuration for a backup server</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>- dump configuration</td>
</tr>
<tr>
<td></td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>- sp_config dump</td>
</tr>
<tr>
<td></td>
<td>- sp_dump history</td>
</tr>
<tr>
<td>manage lock promotion threshold</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>- sp_droplockpromote</td>
</tr>
<tr>
<td></td>
<td>- sp_droprowlockpromote</td>
</tr>
<tr>
<td></td>
<td>- sp_setpglockpromote</td>
</tr>
<tr>
<td></td>
<td>- sp_setrowlockpromote</td>
</tr>
</tbody>
</table>
### 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**

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect*</td>
<td>Connecting to any server using the connect command</td>
</tr>
<tr>
<td>kill</td>
<td>Killing processes owned by the privilege holder</td>
</tr>
<tr>
<td>kill any process</td>
<td>Killing any process owned by any user</td>
</tr>
<tr>
<td>map external file</td>
<td>Mapping a proxy table to a directory or file on a remote server</td>
</tr>
</tbody>
</table>
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations the privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>monitor server replication</td>
<td>Displaying replication status&lt;br&gt;Executing these system procedures:&lt;br&gt;• sp_config_rep_agent (no configure value specified, with or without database name specified)&lt;br&gt;• sp_help_rep_agent (with or without the database name specified)</td>
</tr>
<tr>
<td>set proxy</td>
<td>Executing set proxy to change the identity to another user</td>
</tr>
<tr>
<td>set tracing*</td>
<td>Executing these commands:&lt;br&gt;• set tracefile (for your own session)&lt;br&gt;• set plan for &lt;plan_list&gt; on</td>
</tr>
<tr>
<td>set tracing any process</td>
<td>Executing these commands:&lt;br&gt;• set tracefile (for any session)&lt;br&gt;• set plan for &lt;plan_list&gt; on</td>
</tr>
</tbody>
</table>
Table 8-16: Database-wide privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission management</td>
<td></td>
</tr>
<tr>
<td>manage any object permission</td>
<td>Granting and revoking object permissions</td>
</tr>
<tr>
<td>manage database permissions</td>
<td>Granting and revoking database privileges</td>
</tr>
<tr>
<td>Manage user</td>
<td></td>
</tr>
<tr>
<td>manage any user</td>
<td>Executing these system procedures:</td>
</tr>
<tr>
<td></td>
<td>• sp_addalias</td>
</tr>
<tr>
<td></td>
<td>• sp_addgroup</td>
</tr>
<tr>
<td></td>
<td>• sp_adduser</td>
</tr>
<tr>
<td></td>
<td>• sp_changegroup</td>
</tr>
<tr>
<td></td>
<td>• sp_dropalias</td>
</tr>
<tr>
<td></td>
<td>• sp_dropgroup</td>
</tr>
<tr>
<td></td>
<td>• sp_droptuser</td>
</tr>
</tbody>
</table>

---

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

---
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set user</td>
<td></td>
</tr>
<tr>
<td>setuser</td>
<td>Impersonating another user</td>
</tr>
<tr>
<td>Replication Management</td>
<td></td>
</tr>
<tr>
<td>manage replication</td>
<td>Managing replication settings in a database</td>
</tr>
</tbody>
</table>

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_setrepdump`
- `sp_setrepdbmode`
- `sp_setrepdelfmode`
- `sp_setreplicate`
- `sp_setrepproc`
- `sp_setreptable`
- `sp_start_rep_agent`
- `sp_stop_rep_agent`

Executing these dbcc commands:
- `dbcc gettrunc`
- `dbcc settrunc`
## Maintains database

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
</table>
| 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) |
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>manage database (continued)</td>
<td>Executing these dbcc commands:</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(remap)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(newthreshold)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(findstranded)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(fixlogfreespace)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbr_remap</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(ltmignor)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(updusg_anchors)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(auinlt)</td>
</tr>
<tr>
<td></td>
<td>• dbcc dbrepair(dmap_unlock)</td>
</tr>
<tr>
<td></td>
<td>• dbcc rebuild_text</td>
</tr>
<tr>
<td></td>
<td>• dbcc refreshids (placeobject)</td>
</tr>
<tr>
<td></td>
<td>• dbcc refreshpdes</td>
</tr>
<tr>
<td></td>
<td>• dbcc update_tmode</td>
</tr>
<tr>
<td></td>
<td>• dbcc upgrade_obj</td>
</tr>
<tr>
<td></td>
<td>Executing these built-in functions:</td>
</tr>
<tr>
<td></td>
<td>• derived_stat</td>
</tr>
<tr>
<td></td>
<td>• identity_burn_max</td>
</tr>
<tr>
<td></td>
<td>• lct_admin</td>
</tr>
<tr>
<td></td>
<td>• next_identity</td>
</tr>
</tbody>
</table>
### Privilege name | Operations this privilege authorizes
--- | ---
**Manage query plan** | 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**
- `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
### Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
</table>
| 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_evaluadedb`  
- `sp_dbcc_exclusions`  
- `sp_dbcc_patch_finishtime`  
- `sp_dbcc_updateconfig`                                                                                                                                                                                                                                                                                         |
| 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**

<table>
<thead>
<tr>
<th>select any audit table</th>
<th>Selecting any audit table in <code>sybsecurity</code> (available only in <code>sybsecurity</code> database)</th>
</tr>
</thead>
<tbody>
<tr>
<td>select any system catalog</td>
<td>Selecting all columns from any system table in the current database</td>
</tr>
<tr>
<td>truncate any audit table</td>
<td>Truncating any audit table in <code>sybsecurity</code> (available only in the <code>sybsecurity</code> database)</td>
</tr>
</tbody>
</table>
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

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.

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
Grantable system privileges

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>drop any object</td>
<td>Dropping any of these objects owned by anyone:</td>
</tr>
<tr>
<td></td>
<td>• tables</td>
</tr>
<tr>
<td></td>
<td>• views</td>
</tr>
<tr>
<td></td>
<td>• procedures</td>
</tr>
<tr>
<td></td>
<td>• functions</td>
</tr>
<tr>
<td></td>
<td>• defaults</td>
</tr>
<tr>
<td></td>
<td>• rules</td>
</tr>
<tr>
<td></td>
<td>• indexes</td>
</tr>
<tr>
<td></td>
<td>• triggers</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• drop default</td>
</tr>
<tr>
<td></td>
<td>• drop function</td>
</tr>
<tr>
<td></td>
<td>• drop index</td>
</tr>
<tr>
<td></td>
<td>• drop procedure</td>
</tr>
<tr>
<td></td>
<td>• drop rule</td>
</tr>
<tr>
<td></td>
<td>• drop table</td>
</tr>
<tr>
<td></td>
<td>• drop trigger</td>
</tr>
<tr>
<td></td>
<td>• drop view</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Manage encryption</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>create encryption key*</td>
<td>Creating encryption keys in the database</td>
</tr>
<tr>
<td>manage any encryption key</td>
<td>Creating, altering, and dropping column encryption keys, master keys, and service keys owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• create encryption key</td>
</tr>
<tr>
<td></td>
<td>• alter encryption key</td>
</tr>
<tr>
<td></td>
<td>• drop encryption key</td>
</tr>
<tr>
<td></td>
<td>Executing sp_encryption</td>
</tr>
</tbody>
</table>

| 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                                                    |

<table>
<thead>
<tr>
<th>Defaults</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>create default*</td>
<td>Creating self-owned default</td>
</tr>
<tr>
<td></td>
<td>Execute the create default command</td>
</tr>
</tbody>
</table>

322 Adaptive Server Enterprise
<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>create any default</td>
<td>Creating defaults owned by anyone&lt;br&gt;Execute the create default command</td>
</tr>
<tr>
<td>drop any default</td>
<td>Dropping defaults owned by anyone&lt;br&gt;Execute the drop default command</td>
</tr>
<tr>
<td><strong>Functions</strong></td>
<td></td>
</tr>
<tr>
<td>create function*</td>
<td>Creating self-owned user-defined function&lt;br&gt;Executing these commands:&lt;br&gt;• create function&lt;br&gt;• create function (SQLJ)</td>
</tr>
<tr>
<td>create any function</td>
<td>Creating functions owned by anyone&lt;br&gt;Executing these commands:&lt;br&gt;• create function&lt;br&gt;• create function (SQLJ)</td>
</tr>
<tr>
<td>drop any function</td>
<td>Dropping functions owned by anyone&lt;br&gt;Executing these commands:&lt;br&gt;• drop function&lt;br&gt;• drop function (SQLJ)</td>
</tr>
<tr>
<td>execute any function</td>
<td>Running user-defined functions owned by anyone&lt;br&gt;Execute the execute command</td>
</tr>
<tr>
<td><strong>Indexes</strong></td>
<td></td>
</tr>
<tr>
<td>create any index</td>
<td>Creating indexes on tables owned by anyone&lt;br&gt;Execute the create index command</td>
</tr>
<tr>
<td><strong>Procedures</strong></td>
<td></td>
</tr>
<tr>
<td>create procedure</td>
<td>Creating self-owned procedures&lt;br&gt;Execute the create procedure command</td>
</tr>
<tr>
<td>create any procedure</td>
<td>Creating procedures owned by anyone&lt;br&gt;Execute the create procedure command</td>
</tr>
<tr>
<td>execute any procedure</td>
<td>Execute procedures owned by anyone&lt;br&gt;Execute the execute command.</td>
</tr>
<tr>
<td>drop any procedure</td>
<td>Dropping procedures owned by anyone&lt;br&gt;Execute the drop procedure command</td>
</tr>
<tr>
<td><strong>Rules</strong></td>
<td></td>
</tr>
<tr>
<td>create rule*</td>
<td>Creating self-owned rule&lt;br&gt;Execute the create rule command</td>
</tr>
<tr>
<td>Privilege name</td>
<td>Operations this privilege authorizes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>create any rule</td>
<td>Creating rule owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Execute the create rule command</td>
</tr>
<tr>
<td>drop any rule</td>
<td>Dropping rules owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Execute the drop rule command</td>
</tr>
<tr>
<td>Tables</td>
<td></td>
</tr>
<tr>
<td>alter any table</td>
<td>Altering user tables owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Execute the alter table command</td>
</tr>
<tr>
<td>create any table</td>
<td>Creating user tables owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Execute the create table command</td>
</tr>
<tr>
<td>create table*</td>
<td>Creating self-owned user tables</td>
</tr>
<tr>
<td></td>
<td>Execute the create table command</td>
</tr>
<tr>
<td>decrypt any table</td>
<td>Decrypting any encrypted table</td>
</tr>
<tr>
<td>delete any table</td>
<td>Delete rows of user tables owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• delete table</td>
</tr>
<tr>
<td></td>
<td>• lock table</td>
</tr>
<tr>
<td>drop any table</td>
<td>Dropping user tables owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Execute the drop table command</td>
</tr>
<tr>
<td>identity_insert any table</td>
<td>Enabling or disabling identity_insert on any user table</td>
</tr>
<tr>
<td></td>
<td>Execute the set identity_insert command</td>
</tr>
<tr>
<td>identity_update any table</td>
<td>Enabling or disabling identity_update on any user table</td>
</tr>
<tr>
<td></td>
<td>Execute the set identity_update command</td>
</tr>
<tr>
<td>insert any table</td>
<td>Inserting user tables owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• insert</td>
</tr>
<tr>
<td></td>
<td>• lock table</td>
</tr>
<tr>
<td>manage any statistics</td>
<td>Update or delete statistics on any table owned by anyone</td>
</tr>
<tr>
<td></td>
<td>Executing these commands:</td>
</tr>
<tr>
<td></td>
<td>• delete statistics</td>
</tr>
<tr>
<td></td>
<td>• update statistics</td>
</tr>
<tr>
<td></td>
<td>Executing sp_modifystats</td>
</tr>
<tr>
<td>references any table</td>
<td>Referencing user tables owned by anyone</td>
</tr>
</tbody>
</table>
### Privilege name

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>reorg any table</td>
<td>Reorganizing user tables owned by anyone&lt;br&gt;Execute the <code>reorg</code> command</td>
</tr>
<tr>
<td>select any table</td>
<td>Selecting user tables owned by anyone&lt;br&gt;Execute these commands:&lt;br&gt;• select&lt;br&gt;• lock table (for share lock)&lt;br&gt;• readtext</td>
</tr>
<tr>
<td>transfer any table</td>
<td>Transferring data to or from user tables owned by anyone&lt;br&gt;Execute the <code>transfer table</code> command</td>
</tr>
<tr>
<td>truncate any table</td>
<td>Truncating user tables owned by anyone&lt;br&gt;Execute the <code>truncate table</code> command</td>
</tr>
<tr>
<td>update any table</td>
<td>Updating user tables owned by anyone&lt;br&gt;Execute these commands:&lt;br&gt;• update&lt;br&gt;• lock table&lt;br&gt;• writetext</td>
</tr>
</tbody>
</table>

#### Trigger

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>create trigger*</td>
<td>Creating self-owned trigger.&lt;br&gt;Execute the <code>create trigger</code> command.</td>
</tr>
<tr>
<td>create any trigger</td>
<td>Creating triggers owned by anyone&lt;br&gt;Execute the <code>create trigger</code> command</td>
</tr>
<tr>
<td>drop any trigger</td>
<td>Dropping triggers owned by anyone&lt;br&gt;Execute the <code>drop trigger</code> command</td>
</tr>
</tbody>
</table>

#### Views

<table>
<thead>
<tr>
<th>Privilege name</th>
<th>Operations this privilege authorizes</th>
</tr>
</thead>
<tbody>
<tr>
<td>create view*</td>
<td>Creating self-owned view&lt;br&gt;Execute the <code>create view</code> command.</td>
</tr>
<tr>
<td>create any view</td>
<td>Creating views owned by anyone&lt;br&gt;Execute the <code>create view</code> command.</td>
</tr>
<tr>
<td>drop any view</td>
<td>Dropping views owned by anyone&lt;br&gt;Execute the <code>drop view</code> command.</td>
</tr>
</tbody>
</table>
Grantable system privileges
CHAPTER 9
Confidentiality of Data

Secure Sockets Layer (SSL) in Adaptive Server

Adaptive Server Enterprise security services now 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 Adaptive Server 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.

Internet communications overview

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.
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 imposter.

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** 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.

Adaptive Server 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). Adaptive Server loads its CA trusted root certificate at start-up. When connecting to a remote server for RPCs, Adaptive Server 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).
Secure Sockets Layer (SSL) in Adaptive Server

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 CipherSuite.

For more specific information about the SSL handshake and the SSL/TLS protocol, see the Internet Engineering Task Force Web site at http://www.ietf.org.

For a list of cipher suites that Adaptive Server supports, see “Cipher Suites” on page 340.

SSL in Adaptive Server
Adaptive Server’s 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.
On most platforms, Adaptive Server uses SSL Plus(TM) library API from Certicom Corp. However, for Windows Opteron X64, Adaptive Server uses OpenSSL as the SSL provider.

**SSL filter**

The Adaptive Server 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. Adaptive Server 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 Adaptive Server 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. See “Creating server directory entries” on page 337.

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:

```plaintext
SYBSRV1
 master tcp ether myhostname myport1 ssl
 query tcp ether myhostname myport1 ssl
 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).

**Authentication via the certificate**

The SSL protocol requires server authentication via a server certificate to enable an encrypted session. Likewise, when Adaptive Server 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.
Secure Sockets Layer (SSL) in Adaptive Server

The server certificate

Each Adaptive 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 Adaptive 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`.

Note To make a successful client connection, the common name in the certificate must match the Adaptive Server name in the `interfaces` file.

The CA trusted roots certificate

The list of trusted CAs is loaded by Adaptive Server 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 Adaptive Server. A trusted roots file is accessible by the local Adaptive 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 Adaptive Server 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 Adaptive Server, using a standard ASCII-text editor.

Warning! Use the system security officer role (`sso_role`) within Adaptive Server to restrict access and execution on security-sensitive objects.

Adaptive Server provides tools to generate a certificate request and to authorize certificates. See “Using Adaptive Server tools to request and authorize certificates” on page 336.
Connection types

This section describes various client-to-server and server-to-server connections.

**Client login to Adaptive Server**
Open Client applications establish a socket connection to Adaptive Server 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**
Adaptive Server 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, Adaptive Server requests the remote server’s certificate during connection. Adaptive Server 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 Adaptive Server 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 Adaptive Server use Client-Library to establish connections to servers other than Adaptive Server. The remote server is authenticated by its certificate. The remote server authenticates the Adaptive Server client connection for RPCs with user name and password.

Enabling SSL
Adaptive Server determines which security service it will use for a port based on the interface file (sql.ini on Windows).

- **Enabling SSL**
  1. Generate a certificate for the server.
  2. Create a trusted roots file.
Secure Sockets Layer (SSL) in Adaptive Server

3 Use `sp_configure` to enable SSL. From a command prompt, enter:

```
sp_configure "enable ssl", 1
```

- `1` – enables the SSL subsystem at start-up, allocates memory, and SSL performs wire-level encryption of data across the network.
- `0` (the default) – disables SSL. This value is the default.

4 Add the SSL filter to the `interfaces` file. See “Creating server directory entries” on page 337.

5 Use `sp_ssladmin` to add a certificate to the certificates file. See “Administering certificates” on page 337.

6 Shut down and restart Adaptive Server.

**Note** To request, authorize, and convert third-party certificates, see the *Utility Guide* for information on the `certauth`, `certreq`, and `certpk12` tools.

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 Adaptive Server for SSL connections. The memory is reserved and preallocated within a memory pool and is used internally by Adaptive Server and SSL Plus libraries as requested.

**Obtaining a certificate**

The system security officer installs server certificates and private keys for Adaptive Server by:

- Using third-party tools provided with existing public-key infrastructure already deployed in the customer environment.

- Using the Adaptive Server certificate request tool in conjunction with a trusted third-party CA.
To obtain a certificate, you must request a certificate from a certificate authority (CA). Adaptive Server 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 `certpk12` utility to convert the certificate into a format that is understood by Adaptive Server (see the Utility Guide).

To test the Adaptive Server certificate request tool and to verify that the authentication methods are working on your server, Adaptive Server provides a tool, for testing purposes, that allows you to function as a CA and issue CA-signed certificate to yourself.

The main steps to creating a certificate for use with Adaptive Server 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 Adaptive Server 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 `certpk12` 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`. See the Utility Guide.
Using Adaptive Server tools to request and authorize certificates

Adaptive Server provides two tools for requesting and authorizing certificates. `certreq` generates public and private key pairs and certificate requests. `certauth` converts a server certificate request to a CA-signed certificate.

**Warning!** Use `certauth` only for testing purposes. Sybase recommends that you use the services of a commercial CA because it provides protection for the integrity of the root certificate, and because a certificate that is signed by a widely accepted CA facilitates the migration to the use of client certificates for authentication.

Preparing the server’s trusted root certificate is a five-step process. Perform the first two steps to create a test trusted root certificate so you can verify that you are able to create server certificates. Once you have a test CA certificate (trusted roots certificate) repeat steps three through five to sign server certificates.

1. Use `certreq` to request a certificate.
2. Use `certauth` to convert the certificate request to a CA self-signed certificate (trusted root certificate).
3. Use `certreq` to request a server certificate and private key.
4. Use `certauth` to convert the certificate request to a CA-signed server certificate.
5. Append the private key text to the server certificate and store the certificate in the server’s installation directory.

**Note** Adaptive Server includes the openssl open source utility in `$SYBASE/$SYBASE_OCS/bin`. Use openssl to accomplish all certificate management tasks implemented by `certreq`, `certauth` and `certpk12`. Sybase includes this binary as a convenience, and is not responsible for any issues incurred using the binary. See [www.openssl.org](http://www.openssl.org) for details.

For information about Sybase utilities, `certauth`, `certreq`, and `certpk12` for requesting, authorizing and converting third-party certificates, see the *Utility Guide*.

**Note** `certauth` and `certreq` are dependent on RSA and DSA algorithms. These tools only work with crypto modules that use RSA and DSA algorithms to construct the certificate request.
Creating server directory entries

Adaptive Server accepts client logins and server-to-server RPCs. The address and port numbers where Adaptive Server 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, whereas security mechanisms such as Kerberos are identified with a SECMECH line. The following example shows an entry for an Adaptive Server using SSL in a UNIX environment:

```
SYBSRV1
   master tcp ether myhostname myport ssl
   query tcp ether myhostname myport ssl
   sechmech 1.3.6.1.4.897.4.6.6
```

An entry for the server with SSL and Kerberos security mechanisms on Windows might look like:

```
[SYBSRV1]
   query=nlnwscck, myhostname,myport,ssl
   master=nlnwscck, myhostname,myport,ssl
   sechmech=1.3.6.1.4.897.4.6.6
```

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`

In these examples, the SSL security service is specified on port number `myport`.

**Note** The use of SSL concurrently with a SECMECH security mechanism is intended to facilitate migration from SECMECHs to SSL security.

Administering certificates

To administer SSL and certificates in Adaptive Server, use `sp_ssladmin`. `sso_role` is required to execute the stored procedure.

`sp_ssladmin` is used to:
Secure Sockets Layer (SSL) in Adaptive Server

- 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.

The syntax for `sp_ssladmin` is:

```
sp_ssladmin {[addcert, certificate_path [, password|NULL]]
    [dropcert, certificate_path]
    [lscert]
    [help]}
    [lsciphers]
    [setciphers, {"FIPS" | "Strong" | "Weak" | "All"
        | quoted_list_of_ciphersuites]}
```

For example:

```
sp_ssladmin addcert, "/sybase/ASE-12_5/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-12_5/certificates` (`x:\sybase\ASE-12_5\certificates` on Windows). 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, \texttt{sp\_ssladmin} issues a warning. If the other criteria fails, the certificate is not added to the certificates file.

**Warning!** Adaptive Server limits passwords to 64 characters. In addition, certain platforms restrict the length of valid passwords when creating server certificates. Select a password within these limits:

- Sun Solaris – both 32- and 64-bit platforms, 256 characters.
- Linux – 128 characters.
- IBM – both 32- and 64-bit platforms, 32 characters.
- HP – both 32- and 64-bit platforms, 8 characters.
- Windows – 256 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 \texttt{dataserver} with a \texttt{-y} flag, which prompts the administrator for the private-key password at the command line.

After restarting Adaptive Server with an SSL connection established, use \texttt{sp\_ssladmin} again, this time using the actual password. The password is then encrypted and stored by Adaptive Server. Any subsequent starts of Adaptive 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 Adaptive Server via \texttt{isql}. You can specify “localhost” as the \texttt{hostname} in the \texttt{interfaces} file (\texttt{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.

**Note** Adaptive Server has sufficient memory in its network memory pool to allow \texttt{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, \texttt{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-12_5/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. Adaptive Server needs about 500K bytes of memory
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.

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.

Cipher Suites

During the SSL handshake, the client and server negotiate a common security
protocol via a CipherSuite. **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) organization at

By default, the strongest CipherSuite supported by both the client and the
server is the CipherSuite that is used for the SSL-based session.
Adaptive Server supports the Cipher Suites that are available with the SSL Plus library API and the cryptographic engine, Security Builder™, both from Certicom Corp.

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.

@@ssl_ciphersuite

The Transact-SQL global variable @@ssl_ciphersuite allows users to know which cipher suite was chosen by the SSL handshake and verify that an SSL or a non-SSL connection was established.

Adaptive Server 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.

For example, an isql connection using SSL protocol displays the cipher suite chosen for it.

1> select @@ssl_ciphersuite
2> go

Output:

-----------------------------
TLS_RSA_WITH_AES_128_CBC_SHA
(1 row affected)

Setting SSL cipher suite preferences

In Adaptive Server, sp_ssladmin has two command options to display and set cipher suite preferences: lsciphers and setciphers. With these options, the set of cipher suites that Adaptive Server uses can be restricted, giving control to the system security officer over the kinds of encryption algorithms that may be used by client connections to the server or outbound connections from Adaptive Server. The default behavior for use of SSL cipher suites in Adaptive Server is the same as in earlier versions; it uses an internally defined set of preferences for cipher suites.

To display the values for any set cipher suite preferences, enter:
Secure Sockets Layer (SSL) in Adaptive Server

To set a specific cipher suite preference, enter:

```
sp_ssladmin setciphers, {"FIPS" | "Strong" | "Weak" | "All" | quoted_list_of_ciphersuites }
```

where:

- "FIPS" – is the set of encryptions, hash, and key exchange algorithms that are FIPS-compliant. The algorithms included in this list are AES, 3DES, DES, and SHA1.
- "Strong" – is the set of encryption algorithms using keys longer than 64 bits.
- "Weak" – is the set of encryption algorithms from the set of all supported cipher suites that are not included in the strong set.
- "All" – is the set of default cipher suites.
- quoted_list_of_ciphersuites – specifies a set of cipher suites as a comma-separated list, ordered by preference. Use quotes (") to mark the beginning and end of the list. The quoted list can include any of the predefined sets as well as individual cipher suite names. Unknown cipher suite names cause an error to be reported, and no changes are made to preferences.

The detailed contents of the predefined sets are in Table 9-1 on page 343.

```
sp_ssladmin setciphers
```
sets cipher suite preferences to the given ordered list. This restricts the available SSL cipher suites to the specified set of "FIPS", "Strong", "Weak", "All", or a quoted list of cipher suites. This takes effect on the next listener started, and requires that you restart Adaptive Server to ensure that all listeners use the new settings.

You can display any 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 Adaptive Server uses its default (internal) preferences.
### Table 9-1: Predefined cipher suites in Adaptive Server

<table>
<thead>
<tr>
<th>Set name</th>
<th>Cipher suite names included in the set</th>
</tr>
</thead>
</table>
| FIPS     | TLS_RSA_WITH_AES_256_CBC_SHA  
          | TLS_RSA_WITH_AES_128_CBC_SHA  
          | TLS_RSA_WITH_3DES_EDE_CBC_SHA  
          | TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA  
          | TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA  
          | TLS_RSA_WITH_DES_CBC_SHA  
          | TLS_DHE_DSS_WITH_DES_CBC_SHA  
          | TLS_DHE_RSA_WITH_DES_CBC_SHA  
          | TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA  
          | TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA  |
| Strong   | 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  
          | TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA  
          | TLS_DHE_DSS_WITH_RC4_128_SHA  
          | TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA  |
| Weak     | TLS_RSA_WITH_DES_CBC_SHA  
          | TLS_DHE_DSS_WITH_DES_CBC_SHA  
          | TLS_DHE_RSA_WITH_DES_CBC_SHA  
          | TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA  
          | TLS_RSA_EXPORT1024_WITH_RC4_56_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 |
Table 9-2 describes Cipher suites no longer supported for Adaptive Server 15.0 and later. 15.0. Attempts to use any dropped cipher suite results in an SSLHandshake failure and a failure to connect to Adaptive Server.

<table>
<thead>
<tr>
<th>Set name</th>
<th>Cipher suite names included in the set</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>TLS_RSA_WITH_AES_256_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_AES_128_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_RC4_128_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_RC4_128_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_WITH_RC4_128_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_RSA_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_EXPORT1024_WITH_RC4_56_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_EXPORT_WITH_RC4_40_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
</tbody>
</table>
### Table 9-2: Dropped Cipher suites

<table>
<thead>
<tr>
<th>Set name</th>
<th>Cipher suite names dropped from the set</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPS</td>
<td>TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_RC4_40_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_NULL_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_NULL_SHA</td>
</tr>
<tr>
<td>Weak</td>
<td>TLS_RSA_EXPORT1024_WITH_RC4_56_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_DSA_EXPORT_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td>Others</td>
<td>TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_RC4_40_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_DES_CBC_SHA</td>
</tr>
<tr>
<td></td>
<td>TLS_DH_anon_EXPORT_WITH_NULL_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_NULL_MD5</td>
</tr>
<tr>
<td></td>
<td>TLS_RSA_WITH_NULL_SHA</td>
</tr>
</tbody>
</table>

#### Examples `sp_ssladmin`

On initial startup, before any cipher suite preferences have been set, no preferences are shown by `sp_ssladmin lscipher`.

1> `sp_ssladmin lscipher`
2> `go`

Output:

```
Cipher Suite Name   Preference
------------------  ----------
(0 rows affected)   (return status = 0)
```

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:

<table>
<thead>
<tr>
<th>Cipher Suite Name</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA</td>
<td>1</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA</td>
<td>2</td>
</tr>
<tr>
<td>TLS_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>3</td>
</tr>
<tr>
<td>TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA</td>
<td>4</td>
</tr>
<tr>
<td>TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA</td>
<td>5</td>
</tr>
<tr>
<td>TLS_RSA_WITH_DSS_CBC_SHA</td>
<td>6</td>
</tr>
<tr>
<td>TLS_DHE_DSS_WITH_DSS_CBC_SHA</td>
<td>7</td>
</tr>
</tbody>
</table>
**Secure Sockets Layer (SSL) in Adaptive Server**

<table>
<thead>
<tr>
<th>Cipher Suite Name</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_DHE_RSA_WITH_DES_CBC_SHA</td>
<td>8</td>
</tr>
<tr>
<td>TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA</td>
<td>9</td>
</tr>
<tr>
<td>TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA</td>
<td>10</td>
</tr>
</tbody>
</table>

A preference of 0 (zero) `sp_ssladmin` output indicates a cipher suite is not used by Adaptive Server. The other, non-zero numbers, indicate the preference order that Adaptive Server uses the algorithm 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.

This example uses a quoted list of cipher suites to set preferences in Adaptive Server:

```
1> sp_ssladmin setcipher, '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:

<table>
<thead>
<tr>
<th>Cipher Suite Name</th>
<th>Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA</td>
<td>1</td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA</td>
<td>2</td>
</tr>
</tbody>
</table>

**Other considerations**

When you upgrade to Adaptive Server version 12.5.3 and later, the cipher suite preferences are the server defaults, and `sp_ssladmin` option `lsCipher` displays no preferences. The server uses its default preferences, those defined by "All". 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.

If you upgrade from Adaptive Server version 12.5.3 and later and have set cipher suite preferences, those preferences remain after upgrade. After the upgrade is complete, review your server's cipher suite preferences with current security policies and the lists of supported and unsupported cipher suites found in tables Table 9-1. Omit any cipher suites that are not supported.

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.

Using 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 Adaptive 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.

Specifying 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, Adaptive Server 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:
Kerberos confidentiality

sp_listener 'start','ssltcp:bladel:17251:"CN=ase1.big server 1.com"','0'

See the Reference Manual: Procedures for more information about sp_listener.

**Stored procedure sp_addserver changed**

The filter parameter is enhanced to specify a common name. See the Reference Manual: Procedures.

Kerberos confidentiality

You can also ensure the confidentiality of all messages with Adaptive Server. To require all messages into and out of Adaptive Server 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
```

For more information about using Message Confidentiality with Kerberos and other Security Services supported, see “Administering network-based security” on page 120.

**Dumping and loading 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, Adaptive server issues an error message. If you issue an incorrect password when you attempt to load the database, Adaptive Server issues an error message 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"

Passwords and earlier versions of Adaptive Server

You can use the password-protected dump and load commands only with Adaptive Server version 12.5.2 and later. If you use the password parameter on a dump of a 12.5.2 version of Adaptive Server, the load fails if you try to load it on an earlier version of Adaptive Server.

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 Adaptive Server’s 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.

Adaptive Server version 15.0.2 and later allows you to store portable passwords. See “Character set considerations for passwords” on page 41.
CHAPTER 10 Auditing

This chapter describes how to set up auditing for your installation.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to auditing in Adaptive Server</td>
<td>351</td>
</tr>
<tr>
<td>Installing and setting up auditing</td>
<td>360</td>
</tr>
<tr>
<td>Setting global auditing options</td>
<td>383</td>
</tr>
<tr>
<td>Querying the audit trail</td>
<td>394</td>
</tr>
<tr>
<td>Understanding the audit tables</td>
<td>395</td>
</tr>
</tbody>
</table>

**Note** Permission requirements for operations mentioned in this chapter assume that granular permissions is disabled. Operations may differ when granular permissions is enabled. See Chapter 8, “Using Granular Permissions,” for more information on granular permissions.

Introduction to auditing in Adaptive Server

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 Adaptive Server 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

**Correlating Adaptive Server and operating system audit records**

The easiest way to link Adaptive Server audit records with operating system records is to make Adaptive Server login names the same as operating system login names.

Alternatively, the system security officer can map users’ operating system login names to their Adaptive Server login names. However, this approach requires ongoing maintenance, as login names for new users must be recorded manually.

**The audit system**

The audit system consists of:

- The sybsecurity database, which contains global auditing options and the audit trail
- The in-memory audit queue, to which audit records are sent before they are written to the audit trail
• Configuration parameters for managing auditing
• System procedures for managing auditing

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 Adaptive Server.

The audit trail

Adaptive Server 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. Adaptive Server 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.

Sybase 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.

---

**Warning!** Sybase strongly recommends against using a single audit table on production systems. If you use only a single audit table, you may lose audit records. If you must use only a single audit table because of limited system resources, see “Single-table auditing” on page 379 for instructions.

---

Figure 10-1 shows how the auditing process works with multiple audit tables.
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 Chapter 14, “Archive Database Access,” in the System Administration Guide, Volume 2. For more information about managing the audit trail, see “Setting up audit trail management” on page 370.
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.

**Note** Because audit records are not written directly to the audit trail, you cannot count on an audit record’s being stored immediately in the current audit table.

Auditing configuration parameters

Use these configuration parameters to manage the auditing process:

- **auditing** – enables or disables auditing for the entire Adaptive 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 Adaptive 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`. 
Auditing grant and revoke commands

When auditing the `grant` and `revoke` commands, the command text is written to position two of the `extrainfo` column of the audit record. Comments are filtered unless they are quoted such as: `/*! audit comment */` or bracketed, such as `/*! audit comment */`. Extra white space is filtered. See “Reading the `extrainfo` column” for more information.

If the `grant` or `revoke` command is executed from inside a stored procedure, the grantee and the permission granted will be written instead.

The following is an example of the `extrainfo` column of an audit record for `grant` role `sso_role` to `user1`:

```
sa_role sso_role oper_role sybase_ts_role mon_role;
grant role sso_role to user1; ; ; ; sa/ase;
```

The following is an example of the `extrainfo` column of an audit record for `revoke` role `sso_role` from `user1`:

```
sa_role sso_role oper_role sybase_ts_role mon_role;
revoke role sso_role from user1; ; ; ; sa/ase;
```

Auditing DML statements

For DML statements where predicated privileges are applied, `Predicates Applied: <predicate id>` is written to the `extrainfo` column of the audit record.

The following is an example of an audit record for a `select` statement where two predicated privileges were granted to `user1` on table `t1`:

```
; SELECT; ; ; Predicates Applied: t1_qx8WwJltv#Co,
t1_eb#rxg5pnV76; ; user1/ase;
```

Auditing login and login profile commands

When auditing login and login profile commands, the full text is placed in the `extrainfo` column, with sensitive parameters masked out.

The following is an example of an audit record for a `create login` statement:

```
create login test1 with passwd joe default database master
go
select event,extrainfo from sybsecurity..sysaudits_01 where event=103
go
event extrainfo
```
The following is an example of an audit record for a create login statement with a declare statement for the varchar @pass:

```sql
declare @pass varchar(30)
select @pass = "greatSecret"
create login test3 with passwd @pass default database master
go
select event,extrainfo from sybsecurity..sysaudits_01
where event=103
```

The following is an example of an audit record for a create login statement with an encrypted password:

```sql
create login test4 with encrypted passwd 0xc00749c449a5dd2a59b025c605c80efe26a9235710e18b4ee
db31b32edae356d57a4d86a57388f73c default database master
```

The following is an example of an audit record for an alter login statement modifying the password:

```sql
alter login test1 with passwd joe123 modify passwd myPass123
```
select event, extrainfo from sybsecurity..sysaudits_01 
where event=138 
go 
event extrainfo

-------------------------------------------
138 ; alter login test1 with passwd ****** modify passwd ******; ; ; ; ; test1/ase;

The following is an example of an audit record for the login profile joe_lp:

create login profile joe_lp 
go 
alter login profile joe_lp modify default database 
"sybsystemprocs" 
go 
drop login profile joe_lp 
select event, extrainfo from sybsecurity..sysaudits_01 
where event in (137, 140, 141) 
go 
event extrainfo

-------------------------------------------
138 ; sa_role sso_role oper_role sybase_ts_role; create 
login profile vivekk_lp; ; ; ; ; sa/ase;
140 ; sa_role sso_role oper_role sybase_ts_role; alter 
login profile vivekk_lp modify default database 
"sybsystemprocs"; ; ; ; ; sa/ase;
141 ; sa_role sso_role oper_role sybase_ts_role; drop 
login profile vivekk_lp; ; ; ; ; sa/ase;

Auditing stored procedures

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 section, 
the user name is written as either the owner or the grantee, which ever applies.

The following is an example of an extrainfo column that is owned by the 
database owner, executed by user Joe, and is created with execute as caller:

; EXECUTE AS CALLER; ; ; procedure caller = joe ; ; ;

The following is an example of an extrainfo column for a procedure that is 
owned by Billy and created with execute as owner:
System procedures for auditing

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`.

Managing deployed source in source code control systems

Sybase recommends that all user created source files associated with Adaptive Server 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.

Audit Configuration Changes

Sybase also recommends that changes to Adaptive Server 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.

### Installing and setting up auditing

<table>
<thead>
<tr>
<th>Action and description</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td>“Installing the audit system” on page 360</td>
</tr>
<tr>
<td>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.</td>
<td>“Setting up audit trail management” on page 370 For single-table auditing, “Single-table auditing” on page 379</td>
</tr>
<tr>
<td>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.</td>
<td>“Setting up transaction log management” on page 377</td>
</tr>
<tr>
<td>4. Set auditing options – use sp_audit to establish the events to be audited.</td>
<td>“Setting global auditing options” on page 383</td>
</tr>
<tr>
<td>5. Enable auditing – use sp_configure to turn on the auditing configuration parameter. Adaptive Server begins writing audit records to the current audit table.</td>
<td>“Enabling and disabling auditing” on page 378</td>
</tr>
<tr>
<td>6. Restarting auditing – use sp_audit restart to restart auditing if it fails.</td>
<td>“Restarting auditing” on page 382</td>
</tr>
</tbody>
</table>

### Installing the audit system

There are two methods for installing auditing for the first time in Adaptive Server:

- The auditinit utility – see “Installing auditing with auditinit” on page 362.
• The installsecurity script – see “Installing auditing with installsecurity” on page 367.

Tables and devices for the audit trail

You can specify up to eight system tables (sysaudits_01 through sysaudits_08). Plan to use at least two tables for the audit trail. Put each table on its own device separate from the master device. If you do this, you can use a threshold procedure to automatically archive the current audit table before it fills up and switches to a new empty table for the subsequent audit records.

Device for the syslogs transaction log table

When you install auditing, you must specify a separate device for the transaction log, which consists of the syslogs system table. The syslogs table, which exists in every database, contains a log of the transactions that are executed in the database.

Preinstallation tasks for auditing devices

Sybase recommends that you:

• 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 add more auditing devices later with sp_addauditable. For information, see the Reference Manual: Procedures.

• Install auditing tables and devices in a one-to-one ratio. Tables that share the same device will 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.
Installing and setting up auditing

- 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 (up to eight). 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. Then, you can expand the table size later towards the upper size of the device when a larger auditing capacity is desired, and few or no device additions are available.

- Enable the dsync attribute, or use the directio attribute with that device if you are using a file system device. See the configuration guide for your platform.

Installing auditing with auditinit

- **Configuring Adaptive Server for auditing**
  1. Source SYBASE.csh or SYBASE.sh file if you have not set up the Sybase environment variables.
  2. Start auditinit:
     - UNIX – SYBASE/ASE-15_0/install/auditinit
     - Windows – %SYBASE%/ASE-15_0\install\auditinit

   auditinit displays the following menu:
   
   AUDITINIT
   1. Release directory: /usr/u/sybase
   2. Configure a Server product

   5. Select Configure an Existing Sybase Server.
   6. Select the server to configure.
   7. Provide the SA password for the server you selected.
   8. From the Sybase Server Configuration screen, select Configure Auditing.

   As you proceed through the menus in auditinit, you can change any default values that appear. As you finish each menu, press Ctrl+A to accept the defaults or changed values and move to the next menu.
CONFIGURE AUDITING
1. Configure auditing: no
2. Add a device for audit table(s)
3. Add a device for the audit database transaction log
4. Delete a device entry
5. 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

9 From the Configure Auditing screen, select Configure Auditing.
auditinit re-displays the Configure Auditing menu with the value “yes” displayed for Configure Auditing.

10 Restart Adaptive Server for the changes to take effect.

❖ Creating a device for an audit table

1 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:

2 Select Sybsecurity Physical Device Name.

To create a device for an audit table:

1 Enter the full path of the physical device (file system or raw partition) that you located in “Preinstallation tasks for auditing devices” on page 361.
   Enter the physical name of the device to use for the audit database (default is ""): /dev/path_to_partition
   where path_to_partition is the path to the raw partition or filename for the device.

2 Press Return to acknowledge the warning.
Installing and setting up auditing

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:

3 Proceed through the remaining items on this menu.

Note The Size of the Device value must be equal to or greater than the Device Size for Auditing value. The Device Size for Auditing must be equal to the device size. If you are following Sybase auditing guidelines, you need not change the value displayed in Device Size for Auditing.

4 Press 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. Add a device for audit table(s)
3. Add a device for the audit database transaction log
4. Delete a device entry
5. Change a device entry

List of devices for the audit tables:
Logical name Physical name Segment name Table name Size
Audit_01' secret1/sybase_dr/install/aud1.dat' sysaudits_01 5

5 To add multiple audit devices, repeat steps 1– 6.
You can add as many as eight devices. Sybase 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. Add a device for audit table(s)
3. Add a device for the audit database transaction log
4. Delete a device entry
5. Change a device entry

List of devices for the audit tables:
<table>
<thead>
<tr>
<th>Logical name</th>
<th>Physical name</th>
<th>Segment name</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audit_01'</td>
<td>/secret1/sybase_dr/install/aud1.dat'</td>
<td>sysaudits_01</td>
<td>5</td>
</tr>
<tr>
<td>Audit_02'</td>
<td>/secret1/sybase_dr/install/aud2.dat'</td>
<td>sysaudits_02</td>
<td>5</td>
</tr>
</tbody>
</table>

❖ Creating a device for the audit database transaction log

1. 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:
```

2. 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.

3. Enter the full path name of a physical device.

4. Press Return to acknowledge this warning.

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:
```

5. Proceed through the remaining items on this menu. As you do so, be aware of the following:

- Sybase recommends a minimum size of 2MB for the size of the transaction log.
Installing and setting up auditing

- auditinit displays the size in both 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 the size of the device, based on the assumption that you may want to devote the entire device to log for the auditing task. If you want to use only a subset of the device, you can edit the Size of the Device value.

6 Press Ctrl+A to accept the settings displayed 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. Add a device for audit table(s)
3. Add a device for the audit database transaction log
4. Delete a device entry
5. Change a device entry

List of devices for the audit tables:

<table>
<thead>
<tr>
<th>Logical name</th>
<th>Physical name</th>
<th>Segment name</th>
<th>Table name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>6. Audit_01'</td>
<td>/secret1/sybase_dr/install/aud1.dat'</td>
<td>sysaudits_01</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7. Audit_02'</td>
<td>/secret1/sybase_dr/install/aud2.dat'</td>
<td>sysaudits_02</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>8. auditlog</td>
<td>/secret1/.../auditlog.dat logsegment</td>
<td>syslogs</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

7 When you are ready to execute the audit configuration, press Ctrl+A. auditinit returns you to the Sybase Server Configuration screen.

8 Press Ctrl+A again. auditinit prompts with:

   Execute the Sybase Server Configuration now?

9 Enter “y” (yes).

auditinit executes the tasks to install auditing. When the installation completes successfully, the following messages are displayed:

   Running task: install auditing capabilities.
   ......................Done
   Auditing capability installed.
   Task succeeded: install auditing capabilities.
   Configuration completed successfully.
   Press <return> to continue.
Enabling auditing  After auditing is installed, no auditing occurs until a System Security Officer enables auditing using:

```
sp_configure 'auditing', 1
```

❖  **Deleting a device entry**

1. Select Delete a Device Entry from the Configure Auditing menu.
2. Enter the number of the device to delete.
3. Press return.

❖  **Changing a device entry**

1. Select Change a Device Entry from the Configure Auditing menu.
2. 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. sybase security 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
   ```

3. Select each remaining entry you want to change.
4. Press Ctrl+A to save the new entries.

### Installing auditing with `installsecurity`

The `$SYBASE/ASE-15_0/scripts` directory (`%SYBASE%\ASE-15_0\scripts` on Windows) contains the `installsecurity` script (`instsecu` on Windows) for installing auditing.

**Note** This example assumes a server that uses a logical page size of 2K.

To use `installsecurity` to install auditing:

1. Create the auditing devices and auditing database with the `disk init` and `create database` commands. For example:

   ```
   disk init name = "auditdev",
   physname = "/dev/dsk/c2d0s4",
   ```
Installing and setting up auditing

```plaintext
size = "10M"
disk init name = "auditlogdev",
physname = "/dev/dsk/c2d0s5",
size = "2M"
create database sybsecurity on auditdev
log on auditlogdev
```

2 Use isql to execute the `installsecurity` script:

```plaintext
cd $SYBASE/ASE-12_5/scripts
setenv DSQUERY server_name
isql -Usa -P password -S server_name < installsecurity
```

3 Shut down and restart Adaptive Server.

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_addaudittable`. For information about disk init, create database, and `sp_addaudittable`, see the `Reference Manual: Procedures`.

Moving the auditing database to multiple devices

Place the `sybsecurity` database 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. If you currently have `sybsecurity` on the same device as master, or if you want to move `sybsecurity` to another device, use one of the procedures described in the following sections. When you move the database, you can specify whether to save your existing global audit settings.

Moving `sybsecurity` without saving global audit settings

**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 by following instructions in “Archiving the audit table” on page 371 to avoid losing any historical data that remains in the `sybsecurity` tables.

To move the `sybsecurity` database without saving the global audit settings:
1 Execute the following to remove any information related to logins from the syslogins system table:

   sp_audit "all","all","all","off"

2 Drop the sysbsecurity database.

3 Install sysbsecurity again using the installation procedure described in either:
   - The configuration documentation for your platform, or
   - “Installing auditing with installsecurity” on page 367.

4 During the installation process, place the sysbsecurity database on one or more devices, separate from the master device.

Moving sysbsecurity and saving global audit settings

❖ To move the sysbsecurity database and save the global audit settings

1 Dump the sysbsecurity database:

   dump database sysbsecurity to "/remote/sec_file"

2 Drop the sysbsecurity database:

   drop database sysbsecurity

3 Initialize the first device on which you want to place the sysbsecurity database:

   disk init name = "auditdev",
   physname = "/dev/dsk/c2d0s4",
   size = "10M"

4 Initialize the device where you want to place the security log:

   disk init name = "auditlogdev",
   physname = "/dev/dsk/c2d0s5",
   size = "2M"

5 Create the new sysbsecurity database:

   create database sysbsecurity on auditdev
   log on auditlogdev

6 Load the contents of the old sysbsecurity database into the new database. The global audit settings are preserved:

   load database sysbsecurity from "/remote/sec_file"
Installing and setting up auditing

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.

❖ Creating more than one sysaudits table in sybsecurity

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.

Setting up audit trail management

To effectively manage the audit trail:

1 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.

2 Write a threshold procedure and attach it to each audit table segment.

3 Set configuration parameters for the audit queue size and to indicate appropriate action should the current audit table become full.

The following sections 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, skip to “Single-table auditing” on page 379.

Setting up threshold procedures

Before enabling auditing, establish a threshold procedure to automatically switch auditing tables when the current table is full.

The threshold procedure for the audit device segments should:
• Make the next empty audit table current using \texttt{sp\_configure} to set the current audit table configuration parameter.

• Archive the audit table that is almost full using the insert...select command.

\section*{Changing the current audit table}

The current audit table configuration parameter establishes the table where Adaptive Server writes audit rows. As a system security officer, you can change the current audit table with \texttt{sp\_configure}, using the following syntax, where \textit{n} is an integer that determines the new current audit table:

\begin{verbatim}
sp\_configure "current audit table", n
\end{verbatim}

The valid values for \textit{n} are:

• 1 means \texttt{sysaudits\_01}, 2 means \texttt{sysaudits\_02}, and so forth.

• 0 tells Adaptive Server to automatically set the current audit table to the next table. For example, if your installation has three audit tables, \texttt{sysaudits\_01}, \texttt{sysaudits\_02}, and \texttt{sysaudits\_03}, Adaptive Server sets the current audit table to:
  • 2 if the current audit table is \texttt{sysaudits\_01}
  • 3 if the current audit table is \texttt{sysaudits\_02}
  • 1 if the current audit table is \texttt{sysaudits\_03}

The with truncate option specifies that Adaptive Server should truncate the new table if it is not already empty. If you do not specify this option and the table is not empty, \texttt{sp\_configure} fails.

\textbf{Note} If Adaptive Server truncates the current audit table 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 \texttt{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.

\section*{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.
Be sure that the threshold procedure can successfully copy data into the archive table in another database:

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:

   ```sql
   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`.

   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:

   ```sql
   insert aud_db.sso_user.audit_data
   select * from sybsecurity.dbo.sysaudits_01
   ```

**Example threshold procedure for audit segments**

This sample threshold procedure assumes that three tables are configured for auditing:

```sql
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)

Attaching the threshold procedure to each audit segment

To attach the threshold procedure to each audit table segment, use the
sp_addthreshold.

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

Warning! 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.
Installing and setting up auditing

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 forth. 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
...```

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
```
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 Chapter 16, “Managing Free Space with Thresholds,” in *System Administration Guide: Volume 2*.

**Auditing with the sample threshold procedure in place**

After you enable auditing, Adaptive Server 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, Adaptive Server 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.

**Setting auditing configuration parameters**

Set the following configuration parameters for your auditing installation:

- **audit queue size** sets the number of records in the audit queue in memory.
- **suspend audit when device full** determines what Adaptive Server 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.

**Setting the size of the audit queue**

The default audit queue size is 100 bytes. 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.

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
```
Installing and setting up auditing

For more information about setting the audit queue size and other configuration parameters, see Chapter 5, “Setting Configuration Parameters” in the System Administration Guide: Volume 1.

Suspending 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, Adaptive Server 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), Adaptive Server may truncate the audit table that is full before your threshold procedure has a chance to archive your audit records.
Setting up transaction log management

This section describes guidelines for managing the transaction log in sybsecurity.

If the trunc log on chkpt database option is active, Adaptive Server truncates syslogs every time it performs an automatic checkpoint. After auditing is installed, the value of trunc log on chkpt is on, but you can use sp_dboption to change its value.

Truncating 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. Adaptive Server 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” on page 370, 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 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.

Managing the transaction log with no truncation

If you use db_option to turn the trunc log on chkpt off, the transaction log may fill up. Plan to attach a last-change threshold procedure to the transaction log segment. This procedure gets control when the amount of space remaining on the segment is less than a threshold amount computed automatically by Adaptive Server. 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.

**Note** `sp_modifythreshold` checks to ensure you have “sa_role” active. See “Attaching the threshold procedure to each audit segment” on page 373 for more information.

Adaptive Server does not supply a default procedure, but Chapter 16, “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 `syssecurity` database. You cannot change this option.

With the `trunc log on chkpt` option disable, you can use standard backup and recovery procedures for the `syssecurity` database, but be aware that the audit tables in the restored database may not be in sync with their status during a device failure.

### Enabling and disabling auditing

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
```

**Note** When you enable or disable auditing, Adaptive Server automatically generates an audit record. See event codes 73 and 74 in Table 10-5 on page 397.
Single-table auditing

Sybase 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.

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.

Figure 10-2 shows how the auditing process works with a single audit table.
Establishing and managing single-table auditing

The steps to configure for single-table auditing is the same as for multiple-table auditing.

Threshold procedure for single-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:

```sql
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” on page 373.

---

**Warning!** 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” on page 376.

---

**What happens when the current audit table is full?**

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.

Recovering 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” on page 371. 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.

Restarting auditing

If the audit process is forced to terminate due to an error, sp_audit can be manually restarted by entering:

```
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.
Setting global auditing options

After you have installed auditing, you can use sp_audit to set auditing options. The syntax for sp_audit is:

```
sp_audit option, login_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.

**Note** Auditing does not occur until you activate auditing for the server. For information on how to start auditing, see “Enabling and disabling auditing” on page 378.

Auditing options: types and requirements

The values you can specify for the login_name and object_name parameters to sp_audit depend on the type of auditing option you specify:

- Global options apply to commands that affect the entire server, such as booting the server, disk commands, and allowing ad hoc, user-defined audit records. Option settings for global events are stored in the syssecurity..sysauditoptions system table.
- Database-specific options apply to a database. Examples include altering a database, bulk copy (bcp in) of data into a database, granting or revoking access to objects in a database, and creating objects in a database. Option settings for database-specific events are stored in the master..sysdatabases system table.
- Object-specific options apply to a specific object. Examples include selecting, inserting, updating, or deleting rows of a particular table or view and the execution of a particular trigger or procedure. Option settings for object-specific events are stored in the sysobjects system table in the relevant database.
- User-specific options apply to a specific user or system role. Examples include accesses by a particular user to any table or view or all actions performed when a particular system role, such as sa_role, is active. Option settings for individual users are stored in master..syslogins. The settings for system roles are stored in master..sysauditoptions.
Setting global auditing options

- Role-specific options apply to a specific user, groups, or system roles, and provide fine-grained security-related auditing. The “role” audit option audits all role-related commands, and audit options create, alter, and drop are used to audit role-definition commands, while grant and revoke are used to audit the granting of roles to subjects. The master database is specified for audit options that require an object name parameter.

Table 10-2 shows:

- Valid values for the option and the type of each option – global, database-specific, object-specific, or user-specific
- Valid values for the login_name and object_name parameters for each option
- The database to be in when you set the auditing option
- The command or access that is audited when you set the option
- An example for each option

The default value for all options is off.

Table 10-2: Auditing options, requirements, and examples

<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>adhoc (user-specific)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Allows users to use sp_addauditrecord</td>
</tr>
</tbody>
</table>
|                       |            |             |                                    | This example enables ad hoc user-defined auditing records:  
|                       |            |             |                                    | sp_audit "adhoc", "all", "all", "on" |
| all (user-specific)   | A login name or role | all | Any | All actions of a particular user or by users with a particular role active |
|                       |            |             |                                    | This example turns auditing on for all actions in which the sa_role is active:  
|                       |            |             |                                    | sp_audit "all", "sa_role", "all", "on" |
| alter (database-specific) | all | Database to be audited | Any | alter database, alter role, alter table |
|                       |            |             |                                    | This example turns auditing on for all executions of alter database and alter table in the master database:  
|                       |            |             |                                    | sp_audit @option = "alter", @login_name = "all", @object_name = "master", @setting = "on" |
### Option (option type) | `login_name` | `object_name` | Database to be in to set the option | Command or access being audited
--- | --- | --- | --- | ---
**bcp** (database-specific) | all | Database to be audited | Any | bcp in

This example returns the status of bcp auditing in the pubs2 database:

```sql
sp_audit "bcp", "all", "pubs2"
```

If you do not specify a value for `setting`, Adaptive Server returns the status of auditing for the option you specify.

**bind** (database-specific) | all | Database to be audited | Any | `sp_bindefault`, `sp_bindmsg`, `sp_bindrule`

This example turns bind auditing off for the planning database:

```sql
sp_audit "bind", "all", "planning", "off"
```

**cmdtext** (user-specific) | Login name of the user to be audited | All | Any | SQL text entered by a user.

(Does not reflect whether or not the text in question passed permission checks or not. `eventmod` always has a value of 1.)

This example turns text auditing off for database owners:

```sql
sp_audit "cmdtext", "sa", "all", "off"
```

**create** (database-specific) | all | Database to be audited | Any | create database, create table, create role, create procedure, create trigger, create rule, create default, `sp_addmessage`, create view, create index, create function

**Note** Specify `master` for `object_name` to audit `create database`. You are also auditing the creation of other objects in `master`.

This example turns on auditing of successful object creations in the planning database:

```sql
sp_audit "create", "all", "planning", "pass"
```

The current status of auditing `create database` is not affected because you did not specify the `master` database.

**dbaccess** (database-specific) | all | Database to be audited | Any | Any access to the database from another database

This example audits all external accesses to the project database:

```sql
sp_audit "dbaccess", "all", "project", "on"
```

**dbcc** (global) | all | all | Any | All `dbcc` commands that require permissions

This example audits all executions of the `dbcc` command:

```sql
sp_audit "dbcc", "all", "all", "on"
```
### Setting global auditing options

<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete (object-specific)</td>
<td>all</td>
<td>Name of the table or view to be audited, or default view or default table</td>
<td>The database of the table or view (except tempdb)</td>
<td>delete from a table, delete from a view</td>
</tr>
</tbody>
</table>

This example audits all delete actions for all future tables in the current database:

```
sp_audit "delete", "all", "default table", "on"
```

<table>
<thead>
<tr>
<th>disk (global)</th>
<th>all</th>
<th>all</th>
<th>Any</th>
<th>disk init, disk refit, disk reinit, disk mirror, disk unmirror, disk remirror, disk resize</th>
</tr>
</thead>
</table>

This example audits all disk actions for the server:

```
sp_audit "disk", "all", "all", "on"
```

<table>
<thead>
<tr>
<th>drop (database-specific)</th>
<th>all</th>
<th>Database to be audited</th>
<th>Any</th>
<th>drop database, drop table, drop role, drop procedure, drop index, drop trigger, drop rule, drop default, sp_dropmessage, drop view, drop function</th>
</tr>
</thead>
</table>

This example audits all drop commands in the financial database that fail permission checks:

```
sp_audit "drop", "all", "financial", "fail"
```

<table>
<thead>
<tr>
<th>dump (database-specific)</th>
<th>all</th>
<th>Database to be audited</th>
<th>Any</th>
<th>dump database, dump transaction</th>
</tr>
</thead>
</table>

This example audits dump commands in the pubs2 database:

```
sp_audit "dump", "all", "pubs2", "on"
```

<table>
<thead>
<tr>
<th>encryption_key (database-specific)</th>
<th>all</th>
<th>Database to be audited</th>
<th>Any</th>
<th>alter encryption key, create encryption key, drop encryption key, sp_encryption</th>
</tr>
</thead>
</table>

This example audits all the above commands in the pubs2 database:

```
sp_audit "encryption_key", "all", "pubs2", "on"
```

<table>
<thead>
<tr>
<th>errors (global)</th>
<th>all</th>
<th>all</th>
<th>Any</th>
<th>Fatal error, non-fatal error</th>
</tr>
</thead>
</table>

This example audits errors throughout the server:

```
sp_audit "errors", "all", "all", "on"
```

<table>
<thead>
<tr>
<th>errorlog</th>
<th>all</th>
<th>all</th>
<th>Any</th>
<th>sp_errorlog or the errorlog_admin function</th>
</tr>
</thead>
</table>

This example audits attempts to "change log" to move to a new Adaptive Server error log file:

```
sp_audit "errorlog", "all", "all", "on"
```
<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec_procedure (object-specific)</td>
<td>all</td>
<td>Name of the procedure to be audited or default procedure</td>
<td>The database of the procedure (except tempdb)</td>
<td>execute</td>
</tr>
<tr>
<td>exec_trigger (object-specific)</td>
<td>all</td>
<td>Name of the trigger to be audited or default trigger</td>
<td>The database of the trigger (except tempdb)</td>
<td>Any command that fires the trigger</td>
</tr>
<tr>
<td>func_dbaccess (database-specific)</td>
<td>all</td>
<td>Name of the database you are auditing</td>
<td>Any</td>
<td>Access to the database using the following functions: curunreserved_pgs, db_name, db_id, lct_admin, setdbrepsstat, setrepsstatus, setrepdefmode, is_repagent_enabled, rep_agent_config, rep_agent_admin</td>
</tr>
<tr>
<td>func_obj_access (object-specific)</td>
<td>all</td>
<td>Name of any object that has an entry in sysobjects</td>
<td>Any</td>
<td>Access to an object using the following functions: schema_inc, col_length, col_name, data_pgs, index_col, object_id, object_name, reserved_pgs, rowcnt, used_pgs, has_subquery</td>
</tr>
<tr>
<td>grant (database-specific)</td>
<td>all</td>
<td>Name of the database to be audited</td>
<td>Any</td>
<td>grant</td>
</tr>
</tbody>
</table>

This example turns automatic auditing off for new procedures in the current database:
```
sp_audit "exec_procedure", "all", "default procedure", "off"
```

This example audits all failed executions of the trig_fix_plan trigger in the current database:
```
sp_audit "exec_trigger", "all", "trig_fix_plan", "fail"
```

This example audits accesses to the strategy database via built-in functions:
```
sp_audit @option="func_dbaccess", @login_name="all", @object_name = "strategy", @setting = "on"
```

This example audits accesses to the customer table via built-in functions:
```
sp_audit @option="func_obj_access", @login_name="all", @object_name = "customer", @setting = "on"
```

This example audits all grants in the planning database:
```
sp_audit @option="grant", @login_name="all", @object_name = "planning", @setting = "on"
```
Setting global auditing options

<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>insert (object-specific)</td>
<td>all</td>
<td>Name of the view or table to which you are inserting rows, or default view or default table</td>
<td>The database of the object (except tempdb)</td>
<td>insert into a table, insert into a view</td>
</tr>
<tr>
<td>install (database-specific)</td>
<td>all</td>
<td>Database to be audited</td>
<td>Any</td>
<td>install java</td>
</tr>
<tr>
<td>load (database-specific)</td>
<td>all</td>
<td>Database to be audited</td>
<td>Any</td>
<td>load database, load transaction</td>
</tr>
<tr>
<td>login (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Any login to Adaptive Server</td>
</tr>
<tr>
<td>login_locked (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Any login locked because of exceeding the configured number of failed login attempts</td>
</tr>
<tr>
<td>logout (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Any logout from Adaptive Server</td>
</tr>
<tr>
<td>mount (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>mount database</td>
</tr>
<tr>
<td>password (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Setting of global password and login policy options</td>
</tr>
<tr>
<td>quiesce (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>quiesce database</td>
</tr>
</tbody>
</table>

This example audits all inserts into the dpt_101_view view in the current database:

```
sp_audit "insert", "all", "dpt_101_view", "on"
```

This example audits the installation of java classes in database planning:

```
sp_audit "install", "all", "planning", "on"
```

This example audits all failed executions of database and transaction loads in the projects_db database:

```
sp_audit "load", "all", "projects_db", "fail"
```

This example shows that the login is locked because of exceeding the configured number of failed login attempts:

```
sp_audit "login_locked", "all", "all", "on"
```

This example turns auditing off of logouts from the server:

```
sp_audit "logout", "all", "all", "off"
```

This example audits all mount database commands issued:

```
sp_audit "mount", "all", "all", "on"
```

This example turns auditing on for passwords:

```
sp_audit "password", "all", "all", "on"
```

This example turns auditing on for quiesce database commands:

```
sp_audit "quiesce", "all", "all", "on"
```
<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>all</td>
<td>Name of the view or table to which you are inserting rows, or default view or default table</td>
<td>Any</td>
<td>create table, alter table</td>
</tr>
<tr>
<td>remove</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Audits the removal of Java classes</td>
</tr>
<tr>
<td>revoke</td>
<td>all</td>
<td>Database to be audited</td>
<td>Any</td>
<td>revoke</td>
</tr>
<tr>
<td>rpc</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Remote procedure calls (either in or out)</td>
</tr>
<tr>
<td>security</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Server-wide security-relevant events. See the “security” option in Table 10-5.</td>
</tr>
<tr>
<td>select</td>
<td>all</td>
<td>Name of the view or table to which you are inserting rows, or default view or default table</td>
<td>The database of the object (except tempdb)</td>
<td>select from a table, select from a view</td>
</tr>
<tr>
<td>setuser</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>setuser</td>
</tr>
</tbody>
</table>

This example turns off auditing of the creation of references to the titles table:
```
sp_audit "reference", "all", "titles", "off"
```

This example audits the removal of Java classes in the planning database:
```
sp_audit "remove", "all", "planning", "on"
```

This example turns off auditing of the execution of revoke in the payments_db database:
```
sp_audit "revoke", "all", "payments_db", "off"
```

This example audits all remote procedure calls out of or into the server:
```
sp_audit "rpc", "all", "all", "on"
```

This example audits server-wide security-relevant events in the server:
```
sp_audit "security", "all", "all", "on"
```

This example audits all failed selects from the customer table in the current database:
```
sp_audit "select", "all", "customer", "fail"
```

This example audits all executions of setuser in the projdb database:
```
sp_audit "setuser", "all", "projdb", "on"
```
### Setting global auditing options

<table>
<thead>
<tr>
<th>Option (option type)</th>
<th>login_name</th>
<th>object_name</th>
<th>Database to be in to set the option</th>
<th>Command or access being audited</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_access (user-specific)</td>
<td>all</td>
<td>Any</td>
<td>select, delete, update, or insert access in a table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Login name of the user to be audited.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This example audits all table accesses by the login named “smithson”:</td>
<td></td>
<td><code>sp_audit &quot;table_access&quot;, &quot;smithson&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td>transfer_table (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>Server-wide option. Does not appear in <code>sysauditoptions</code>.</td>
</tr>
<tr>
<td></td>
<td>This example audits server-wide transfer-relevant events in the server:</td>
<td></td>
<td><code>sp_audit &quot;transfer_table&quot;, &quot;tdb1.table1&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td>truncate (database-specific)</td>
<td>all</td>
<td>Database to be audited</td>
<td>Any</td>
<td>truncate table</td>
</tr>
<tr>
<td></td>
<td>This example audits all table truncations in the <code>customer</code> database:</td>
<td></td>
<td><code>sp_audit &quot;truncate&quot;, &quot;all&quot;, &quot;customer&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td>unbind (database-specific)</td>
<td>all</td>
<td>Database to be audited</td>
<td>Any</td>
<td><code>sp_unbindefault</code>, <code>sp_unbindrule</code>, <code>sp_unbindmsg</code></td>
</tr>
<tr>
<td></td>
<td>This example audits all failed attempts of unbinding in the <code>master</code> database:</td>
<td></td>
<td><code>sp_audit &quot;unbind&quot;, &quot;all&quot;, &quot;master&quot;, &quot;fail&quot;</code></td>
<td></td>
</tr>
<tr>
<td>unmount (global)</td>
<td>all</td>
<td>all</td>
<td>Any</td>
<td>unmount database</td>
</tr>
<tr>
<td></td>
<td>This example audits all attempts to unmount or create a manifest file with any database:</td>
<td></td>
<td><code>sp_audit &quot;unmount&quot;, &quot;all&quot;, &quot;all&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td>update (object-specific)</td>
<td>all</td>
<td>Name specifying the object to be audited, default table or default view</td>
<td>The database of the object (except <code>tempdb</code>)</td>
<td>update to a table, update to a view</td>
</tr>
<tr>
<td></td>
<td>This example audits all attempts by users to update the <code>projects</code> table in the current database:</td>
<td></td>
<td><code>sp_audit &quot;update&quot;, &quot;all&quot;, &quot;projects&quot;, &quot;on&quot;</code></td>
<td></td>
</tr>
<tr>
<td>view_access (user-specific)</td>
<td>Login name all</td>
<td>Any</td>
<td>select, delete, insert, or update to a view</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This example turns off view auditing of user “joe”:</td>
<td></td>
<td><code>sp_audit &quot;view_access&quot;, &quot;joe&quot;, &quot;all&quot;, &quot;off&quot;</code></td>
<td></td>
</tr>
</tbody>
</table>
Examples of setting auditing options

Suppose you want to audit all failed deletions on the projects table in the company_operations database and for all new tables in the database. Use the object-specific delete option for the projects table and use default table for all future tables in the database. You must be in the object’s database before you execute sp_audit to set object-specific auditing options:

```
sp_audit "security", "all", "all", "fail"
```

For this example, execute:

```
use company_operations

sp_audit "delete", "all", "projects", "fail"
sp_audit "delete", "all", "default table", "fail"
```

| Role definition auditing | Example 1 | Turns on auditing for role alterations:
| | sp_audit "alter", "all", "master", "pass"
| Example 2 | Turns on auditing for successful role creations:
| | sp_audit "alter", "all", "master", "on"
| Example 3 | This example turns off auditing of dropping roles:
| | sp_audit "drop", "all", "master", "off"
| Example 4 | Turns off auditing of granting roles:
| | sp_audit "grant", "all", "master", "off"

Auditing is performed using the grant or role audit option generating the AUD_EVT_UDR_CMD (85) event audit record.

Example 5  Turns on auditing of revoking rules:

```
sp_audit "revoke", "all", "master", "on"
```

Auditing is performed using the revoke or role audit option generating the AUD_EVT_UDR_CMD (85) event audit record.
Setting global auditing options

Hiding system stored procedure and command password parameters

When auditing is configured and enabled, and the `sp_audit` option `cmdtext` is set, system stored procedure and command password parameters are replaced with a fixed length string of asterisks in the audit records contained in the audit logs.

For example, execute the following when auditing is enabled and `sp_audit cmdtext` is set:

```
alter login johnd with password oldpasswd modify
password 'newpasswd'
```

The command results in output similar to:

```
alter login johnd with password ****** modify password ******
```

An example of the stored procedure `sp_addlogin` when auditing is enabled:

```
sp_addlogin test2,secret
```

```
go
select event,extrainfo from sybsecurity..sysaudits_01
where event = 92
```

```
go
```

The auditing record results in the output:

```
event extrainfo
-------------------------------------------
92 ; sp_addlogin test2,******
```

This protects passwords from being seen by other with access to the audit log.

Determining 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 [, name]]
```

For more information, see `sp_displayaudit` in Reference Manual: Procedures.
Adding user-specified records to the audit trail

`sp_addauditrecord` allows users to enter comments into the audit trail. The syntax is:

```sql
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.

### Examples of adding user-defined audit records

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:
Querying the audit trail

To query the audit trail, use SQL to select and summarize the audit data. If you follow the procedures discussed in “Setting up audit trail management” on page 370, 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:

```sql
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:

```sql
select * from audit_data
where extrainfo like "%sso_role%"
and dbname = "pubs2"
GO
```

This command requests audit records for all table truncations (event 64):

```sql
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:

```sql
select * from audit_data where audit_event_name(event) = "Create Database"
go
```

### Understanding the 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`.

Table 10-3 describes the columns in all audit tables.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>event</td>
<td>smallint</td>
<td>Type of event being audited. See Table 10-5 on page 397.</td>
</tr>
</tbody>
</table>
| eventmod    | smallint | More information about the event being audited. Indicates whether or not the event in question passed permission checks. Possible values are:  
  • 0 = no modifier for this event.  
  • 1 = the event passed permission checking.  
  • 2 = the event failed permission checking. |
| spid        | smallint | ID of the process that caused the audit record to be written. |
| eventtime   | datetime | Date and time that the audited event occurred. |
| sequence    | smallint | Sequence number of the record within a single event. Some events require more than one audit record. |
| suid        | smallint | Server login ID of the user who performed the audited event. |
| dbid        | int null | Database ID in which the audited event occurred, or in which the object, stored procedure, or trigger resides, depending on the type of event. |
| objid       | int null | ID of the accessed object, stored procedure, or trigger. |
| xactid      | binary(6) null | ID of the transaction containing the audited event. For a multi-database transaction, this is the transaction ID from the database where the transaction originated. |
| loginname   | varchar(30) null | Login name corresponding to the suid. |
| dbname      | varchar(30) null | Database name corresponding to the dbid. |
| objname     | varchar(30) null | Object name corresponding to the objid. |
| objowner    | varchar(30) null | Name of the owner of objid. |
Understanding the audit tables

<table>
<thead>
<tr>
<th>Column name</th>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>extrainfo</td>
<td>varchar(255) null</td>
<td>Additional information about the audited event. This column contains a sequence of items separated by semicolons. For details, see “Reading the extrainfo column” on page 396.</td>
</tr>
<tr>
<td>nodeid</td>
<td>tinyint</td>
<td>Server nodeid in a cluster where the event occurred.</td>
</tr>
</tbody>
</table>

Reading the *extrainfo* column

The *extrainfo* column contains a sequence of data separated by semicolons. The data is organized in the following categories.

<table>
<thead>
<tr>
<th>Position</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Roles</td>
<td>A list of active roles, separated by blanks.</td>
</tr>
<tr>
<td>2</td>
<td>Keywords or Options</td>
<td>The name of the keyword or option that was used for the event. For example, for the <em>alter table</em> command, the <em>add column</em> or <em>drop constraint</em> options might have been used. If multiple keywords or options are listed, they are separated by commas.</td>
</tr>
<tr>
<td>3</td>
<td>Previous value</td>
<td>If the event resulted in the update of a value, this item contains the value prior to the update.</td>
</tr>
<tr>
<td>4</td>
<td>Current value</td>
<td>If the event resulted in the update of a value, this item contains the new value.</td>
</tr>
<tr>
<td>5</td>
<td>Other information</td>
<td>Additional security-relevant information that is recorded for the event.</td>
</tr>
<tr>
<td>6</td>
<td>Proxy information</td>
<td>The original login name if the event occurred while a set proxy was in effect.</td>
</tr>
<tr>
<td>7</td>
<td>Principal name</td>
<td>The principal name from the underlying security mechanism if the user’s login is the secure default login, and the user logged in to Adaptive Server via unified login. The value of this item is NULL if the secure default login is not being used.</td>
</tr>
</tbody>
</table>

This example shows an extrainfo column entry for the event of changing an auditing configuration parameter.

```
so_role; suspend audit when device full; 1; 0; ; ralph;
```

This entry indicates that a system security officer changed *suspend audit when device full* from 1 to 0. There is no “other information” for this entry. The sixth category indicates that the user “ralph” was operating with a proxy login. No principal name is provided.

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).
Table 10-5 lists the values that appear in the event column, arranged by `sp_audit` option. The “Information in extrainfo” column describes information that might appear in the extrainfo column of an audit table, based on the categories described in Table 10-4.

**Table 10-5: Values in event and extrainfo columns**

<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Automatically audited event not controlled by an option)</td>
<td>Enabling auditing with: <code>sp_configure auditing</code></td>
<td>73</td>
<td>—</td>
</tr>
<tr>
<td>(Automatically audited event not controlled by an option)</td>
<td>Disabling auditing with: <code>sp_configure auditing</code></td>
<td>74</td>
<td>—</td>
</tr>
<tr>
<td>Unlocking Administrator’s account</td>
<td>Disabling auditing with: <code>sp_configure auditing</code></td>
<td>74</td>
<td>—</td>
</tr>
<tr>
<td>adhoc</td>
<td>User-defined audit record</td>
<td>1</td>
<td><code>extrainfo</code> is filled by the text parameter of <code>sp_addauditrecord</code></td>
</tr>
</tbody>
</table>
## Understanding the audit tables

<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>alter</td>
<td>alter database</td>
<td>2</td>
<td><strong>Subcommand keywords:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- alter maxhold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- alter size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- inmemory</td>
</tr>
<tr>
<td>alter...modify owner name_in_db</td>
<td></td>
<td>124</td>
<td><strong>Subcommand keywords:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For user-defined types: owner.obj_name name_in_db preserve permissions if the option is specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For objects: name_in_db preserve permission if the option is specified.</td>
</tr>
<tr>
<td>alter...modify owner login_name</td>
<td></td>
<td>124</td>
<td><strong>Subcommand keywords:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Do not apply to user-defined datatypes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- For objects: login_name preserve permissions if the option is specified.</td>
</tr>
<tr>
<td>alter table</td>
<td></td>
<td>3</td>
<td><strong>Subcommand keywords:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- add/drop/modify column</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- replace columns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- replace decrypt default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- replace/add decrypt default</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- add constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- drop constraint</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If one or more encrypted columns are added, extrainfo contains the following, where keyname is the fully qualified name of the key: add/drop/modify column column1/keyname1, [column2/keyname2]</td>
</tr>
<tr>
<td>bcp</td>
<td>bcp in</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>bind</td>
<td>sp_binddefault</td>
<td>6</td>
<td><strong>Other information:</strong> Name of the default</td>
</tr>
<tr>
<td></td>
<td>sp_bindmsg</td>
<td>7</td>
<td><strong>Other information:</strong> Message ID</td>
</tr>
<tr>
<td></td>
<td>sp_bindrule</td>
<td>8</td>
<td><strong>Other information:</strong> Name of the rule</td>
</tr>
<tr>
<td>all, create</td>
<td>create database</td>
<td>9</td>
<td>Keywords or options: inmemory</td>
</tr>
<tr>
<td>cmdtext</td>
<td>All commands</td>
<td>92</td>
<td>Full text of command, as sent by the client</td>
</tr>
</tbody>
</table>
### Audit option | Command or access to be audited | event | Information in extrainfo
--- | --- | --- | ---
create | create database | 9 | —
create default | 14 | —
create procedure | 11 | —
create rule | 13 | —
create table | 10 | For encrypted columns, extrainfo contains column names and keynames. EK column1/keyname1[,column2 keyname2] where EK is a prefix indicating that subsequent information refers to encryption keys and keyname is the fully qualified name of the key.
create trigger | 12 | —
create view | 16 | —
create index | 104 | Other information: Name of the index
create function | 97 | —
sp_addmessage | 15 | Other information: Message number
dbaccess | Any access to the database by any user | 17 | Keywords or options: use cmd outside reference
dbcc | dbcc all keywords | 81 | Keywords or options: Any of the dbcc keywords such as checkstorage and the options for that keyword.
delete | delete from a table | 18 | Keywords or options: delete
delete from a view | 19 | Keywords or options: delete
<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
</table>
| disk         | disk init                       | 20    | *Keywords or options:* disk init  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk mirror                     | 23    | *Keywords or options:* disk mirror  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk refit                      | 21    | *Keywords or options:* disk refit  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk reinit                     | 22    | *Keywords or options:* disk reinit  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk release                    | 87    | *Keywords or options:* disk release  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk remirror                   | 25    | *Keywords or options:* disk remirror  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk unmirror                   | 24    | *Keywords or options:* disk unmirror  
|              |                                 |       | *Other information:* Name of the disk |
| disk         | disk resize                     | 100   | *Keywords or options:* disk resize  
|              |                                 |       | *Other information:* Name of the disk |
| drop         | drop database                   | 26    | —                         |
| drop         | drop default                    | 31    | —                         |
| drop         | drop procedure                  | 28    | —                         |
| drop         | drop table                      | 27    | —                         |
| drop         | drop trigger                    | 29    | —                         |
| drop         | drop rule                       | 30    | —                         |
| drop         | drop view                       | 33    | —                         |
| drop         | drop index                      | 105   | *Other information:* Index name |
| drop         | drop function                   | 98    | —                         |
|              | sp_dropmessage                  | 32    | *Other information:* Message number |
| dump         | dump database                   | 34    | —                         |
| dump         | dump transaction                | 35    | —                         |
| encryption_key | sp_encryption                   | 106   | *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 ******** ********`
<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>create encryption key</td>
<td>107</td>
<td></td>
<td>Keywords contain: algorithm name-bitlength/IV [random/NULL]/pad [random</td>
</tr>
<tr>
<td>alter encryption key</td>
<td>108</td>
<td>default/not default</td>
<td></td>
</tr>
<tr>
<td>drop encryption key</td>
<td>109</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| AEK modify encryption | 118                             | modify encryption with user passwd | for user username 
{with login passwd 
  with user passwd 
  with keyvalue} 
  for recovery Note that keyvalue is displayed only for replication of alter encryption key modify encryption. For example, when user “stephen” modifies his key copy, the following information is saved: MODIFY ENCRYPTION for user stephen WITH USER PASSWD |
| AEK add encryption     | 119                             | add encryption for user user_name for login association | recovery | with keyvalue Note that keyvalue is displayed only for replication of alter encryption key add encryption. |
| alter encryption key drop encryption | 120                  | drop encryption [for recovery | for user user_name See the Encrypted Columns Users Guide. |
| alter encryption key modify owner     | 121                             | modify owner [new owner user_name] See the Encrypted Columns Users Guide. |
| alter encryption key recover key     | 122                             | recovery key [with key_value] with keyvalue is only used during replication of alter encryption key See the Encrypted Columns Users Guide. |
| errorlog or errorlog_admin function | 127                             | The parameters passed to errorlog_admin are logged to identify the subcommand: errorlog_admin (param1, param2,...). |
## Understanding the audit tables

<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>errors</td>
<td>Fatal error</td>
<td>36</td>
<td>Other information: Error number, Severity, State</td>
</tr>
<tr>
<td></td>
<td>Non-fatal error</td>
<td>37</td>
<td>Other information: Error number, Severity, State</td>
</tr>
<tr>
<td>exec_procedure</td>
<td>Execution of a procedure</td>
<td>38</td>
<td>Other information: All input parameters</td>
</tr>
<tr>
<td>exec_trigger</td>
<td>Execution of a trigger</td>
<td>39</td>
<td>—</td>
</tr>
<tr>
<td>func_obj_access,</td>
<td>Accesses to objects and databases via Transact-SQL</td>
<td>86</td>
<td>—</td>
</tr>
<tr>
<td>func_dbaccess</td>
<td>functions. (Auditing must be enabled for the sa_role to audit functions).</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>grant</td>
<td>grant</td>
<td>40</td>
<td>Contains the full command text if available. Otherwise, contains the grantee and command type.</td>
</tr>
<tr>
<td>insert</td>
<td>insert into a table</td>
<td>41</td>
<td>Keywords or option:</td>
</tr>
<tr>
<td></td>
<td>• If insert is used: insert</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• If select into is used: insert into followed by the fully qualified object name</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>insert into a view</td>
<td>42</td>
<td>Keywords or options: insert</td>
</tr>
<tr>
<td>install</td>
<td>install</td>
<td>93</td>
<td>—</td>
</tr>
<tr>
<td>load</td>
<td>load database</td>
<td>43</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>load transaction</td>
<td>44</td>
<td>—</td>
</tr>
<tr>
<td>login</td>
<td>Any login to the server</td>
<td>45</td>
<td>Other information:</td>
</tr>
<tr>
<td></td>
<td>• Host name and IP address of the machine</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Error number, Severity, State for failed logins.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>login_locked</td>
<td>Login locked due to exceeding the configured number of failed login attempts</td>
<td>112</td>
<td>—</td>
</tr>
<tr>
<td>logout</td>
<td>Any logouts from the server</td>
<td>46</td>
<td>Other information: Host name</td>
</tr>
<tr>
<td>mount</td>
<td>mount database</td>
<td>101</td>
<td>—</td>
</tr>
<tr>
<td>password</td>
<td>sp_passwordpolicy and all its actions except list.</td>
<td>115</td>
<td>Parameters for sp_passwordpolicy</td>
</tr>
<tr>
<td>quiesce</td>
<td>quiesce database</td>
<td>96</td>
<td>—</td>
</tr>
<tr>
<td>reference</td>
<td>Creation of references to tables</td>
<td>91</td>
<td>Keywords or options: reference</td>
</tr>
<tr>
<td></td>
<td>• Other information: Name of the referencing table</td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td>remove java</td>
<td>94</td>
<td>—</td>
</tr>
<tr>
<td>Audit option</td>
<td>Command or access to be audited</td>
<td>event</td>
<td>Information in extrainfo</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------</td>
<td>-------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>revoke</td>
<td>revoke</td>
<td>47</td>
<td>Contains the full command text if available. Otherwise, contains the grantee and command type.</td>
</tr>
<tr>
<td>rpc</td>
<td>Remote procedure call from another server</td>
<td>48</td>
<td>Keywords or options: Name of client program Other information: Server name, host name of the machine from which the RPC was executed.</td>
</tr>
<tr>
<td>role locked</td>
<td>Role setting/unsetting</td>
<td>133</td>
<td>Role name and lock reason:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Role locked by suid by manually executing alter role rolename lock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Role locked by Adaptive Server due to failed role activation attempts reaching max failed_logins</td>
</tr>
<tr>
<td>security</td>
<td>connect to (CIS only)</td>
<td>90</td>
<td>Keywords or options: connect to</td>
</tr>
<tr>
<td></td>
<td>online database</td>
<td>83</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>proc_role function (executed from within a system procedure)</td>
<td>80</td>
<td>Other information: Required roles</td>
</tr>
<tr>
<td></td>
<td>Regeneration of a password by an sso</td>
<td>76</td>
<td>Keywords or options: Setting SSO password Other information: Login name</td>
</tr>
<tr>
<td>role toggling</td>
<td></td>
<td>55</td>
<td>Previous value: on or off Current value: on or off Other information: Name of the role being set</td>
</tr>
<tr>
<td>Server start</td>
<td></td>
<td>50</td>
<td>Other information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-d masterdevicename</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-i interfaces file path</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-S servername</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-e errorfilename</td>
</tr>
<tr>
<td>sp_webservices</td>
<td></td>
<td>111</td>
<td>Keywords or options: deploy if deploying a web service. deploy_all if deploying all web services</td>
</tr>
<tr>
<td>sp_webservices</td>
<td></td>
<td>111</td>
<td>Keywords or options: undeploy if undeploying a web service. undeploy_all if undeploying all web services</td>
</tr>
<tr>
<td>Server shutdown</td>
<td></td>
<td>51</td>
<td>Keywords or options: shutdown</td>
</tr>
<tr>
<td>set proxy or set session authorization</td>
<td></td>
<td>88</td>
<td>Previous value: Previous suid Current value: New suid</td>
</tr>
<tr>
<td>Audit option</td>
<td>Command or access to be audited</td>
<td>event</td>
<td>Information in extrainfo</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
<td>-------</td>
<td>-------------------------</td>
</tr>
</tbody>
</table>
| sp_configure | 82 | *Keywords or options:* SETCONFIG  
*Other information:*  
- 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_ssladmin administration enabled | 99 | Keywords contains SSL_ADMIN addcert, if adding a certification. |
| Audit table access | 61 | — |
| create login, drop login | 103 | *Keywords or options:* create login, drop login |
| create, drop, alter, grant, or revoke role | 85 | *Keywords or options:* create, drop, alter, grant, or revoke role |
| built-in functions | 86 | *Keywords or options:* Name of function |
| Security command or access to be audited, specifically, starting Adaptive Server with -u option to unlock the administrator’s account. | 95 | Other information contains ‘Unlocking admin account’ |
| Changes to the LDAP state changes | 123 | *Keywords or options:* Primary URL state and secondary URL state  
- Previous value  
- Current value  
Additional information indicates whether the state change happened automatically or because of a manually entered command. |
| The regeneration of asymmetric keypairs for network password encryption by the system or sp_passwordpolicy | 117 | Information in extrainfo |
| select | select from a table | 62 | *Keywords or options:* select into select readtext |
| select from a view | 63 | *Keywords or options:* select into select readtext |
| setuser | setuser | 84 | *Other information:* Name of the user being set |
Table 10-6 lists the values that appear in the event column, arranged by the audit event.

<table>
<thead>
<tr>
<th>Audit option</th>
<th>Command or access to be audited</th>
<th>event</th>
<th>Information in extrainfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_access</td>
<td>delete</td>
<td>18</td>
<td>Keywords or options: delete</td>
</tr>
<tr>
<td></td>
<td>insert</td>
<td>41</td>
<td>Keywords or options: insert</td>
</tr>
<tr>
<td></td>
<td>select</td>
<td>62</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>select into</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>readtext</td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>70</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>writetext</td>
</tr>
<tr>
<td>truncate</td>
<td>truncate table</td>
<td>64</td>
<td>—</td>
</tr>
<tr>
<td>transfer_table</td>
<td>transfer table</td>
<td>136</td>
<td>transfer table</td>
</tr>
<tr>
<td>unbind</td>
<td>sp_unbindefault</td>
<td>67</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>sp_unbindmsg</td>
<td>69</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>sp_unbindrule</td>
<td>68</td>
<td>—</td>
</tr>
<tr>
<td>unmount</td>
<td>unmount database</td>
<td>102</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>create manifest file</td>
<td>116</td>
<td>Information in extrainfo</td>
</tr>
<tr>
<td>update</td>
<td>update to a table</td>
<td>70</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>writetext</td>
</tr>
<tr>
<td></td>
<td>update to a view</td>
<td>71</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>writetext</td>
</tr>
<tr>
<td>view_access</td>
<td>delete</td>
<td>19</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td>insert</td>
<td>42</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>select into</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>readtext</td>
</tr>
<tr>
<td></td>
<td>update</td>
<td>71</td>
<td>Keywords or options:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>update</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>writetext</td>
</tr>
</tbody>
</table>

Table 10-6: Audit event values

<table>
<thead>
<tr>
<th>Audit event ID</th>
<th>Command name</th>
<th>Audit event ID</th>
<th>Command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ad hoc audit record</td>
<td>62</td>
<td>select table</td>
</tr>
</tbody>
</table>
### Understanding the audit tables

<table>
<thead>
<tr>
<th>Audit event ID</th>
<th>Command name</th>
<th>Audit event ID</th>
<th>Command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>alter database</td>
<td>68</td>
<td>unbind rule</td>
</tr>
<tr>
<td>3</td>
<td>alter table</td>
<td>69</td>
<td>unbind message</td>
</tr>
<tr>
<td>4</td>
<td>bcp in</td>
<td>70</td>
<td>update table</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>71</td>
<td>update view</td>
</tr>
<tr>
<td>6</td>
<td>bind default</td>
<td>72</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>bind message</td>
<td>73</td>
<td>auditing enabled</td>
</tr>
<tr>
<td>8</td>
<td>bind rule</td>
<td>74</td>
<td>auditing disabled</td>
</tr>
<tr>
<td>9</td>
<td>create database</td>
<td>75</td>
<td>Reserved</td>
</tr>
<tr>
<td>10</td>
<td>create table</td>
<td>76</td>
<td>SSO changed password</td>
</tr>
<tr>
<td>11</td>
<td>create procedure</td>
<td>77</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>create trigger</td>
<td>78</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>create rule</td>
<td>79</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>create default</td>
<td>80</td>
<td>role check performed</td>
</tr>
<tr>
<td>15</td>
<td>create message</td>
<td>81</td>
<td>dbcc</td>
</tr>
<tr>
<td>16</td>
<td>create view</td>
<td>82</td>
<td>config</td>
</tr>
<tr>
<td>17</td>
<td>access to database</td>
<td>83</td>
<td>online database</td>
</tr>
<tr>
<td>18</td>
<td>delete table</td>
<td>84</td>
<td>setuser command</td>
</tr>
<tr>
<td>19</td>
<td>delete view</td>
<td>85</td>
<td>create role, drop role, alter role, grant role, or revoke role</td>
</tr>
<tr>
<td>20</td>
<td>disk init</td>
<td>86</td>
<td>built-in function</td>
</tr>
<tr>
<td>21</td>
<td>disk refit</td>
<td>87</td>
<td>Disk release</td>
</tr>
<tr>
<td>22</td>
<td>disk reinit</td>
<td>88</td>
<td>set SSA command</td>
</tr>
<tr>
<td>23</td>
<td>disk mirror</td>
<td>89</td>
<td>kill or terminate command</td>
</tr>
<tr>
<td>24</td>
<td>disk unmirror</td>
<td>90</td>
<td>connect</td>
</tr>
<tr>
<td>25</td>
<td>disk remirror</td>
<td>91</td>
<td>reference</td>
</tr>
<tr>
<td>26</td>
<td>drop database</td>
<td>92</td>
<td>command text</td>
</tr>
<tr>
<td>27</td>
<td>drop table</td>
<td>93</td>
<td>JCS install command</td>
</tr>
<tr>
<td>28</td>
<td>drop procedure</td>
<td>94</td>
<td>JCS remove command</td>
</tr>
<tr>
<td>29</td>
<td>drop trigger</td>
<td>95</td>
<td>Unlock admin account</td>
</tr>
<tr>
<td>30</td>
<td>drop rule</td>
<td>96</td>
<td>quiesce database</td>
</tr>
<tr>
<td>31</td>
<td>drop default</td>
<td>97</td>
<td>create SQLJ function</td>
</tr>
<tr>
<td>32</td>
<td>drop message</td>
<td>98</td>
<td>drop SQLJ function</td>
</tr>
<tr>
<td>33</td>
<td>drop view</td>
<td>99</td>
<td>SSL administration</td>
</tr>
<tr>
<td>34</td>
<td>dump database</td>
<td>100</td>
<td>disk resize</td>
</tr>
<tr>
<td>35</td>
<td>dump transaction</td>
<td>101</td>
<td>mount database</td>
</tr>
<tr>
<td>36</td>
<td>Fatal error</td>
<td>102</td>
<td>unmount database</td>
</tr>
<tr>
<td>37</td>
<td>Non-fatal error</td>
<td>103</td>
<td>create login</td>
</tr>
<tr>
<td>Audit event ID</td>
<td>Command name</td>
<td>Audit event ID</td>
<td>Command name</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>38</td>
<td>execution of stored procedure</td>
<td>104</td>
<td>create index</td>
</tr>
<tr>
<td>39</td>
<td>Execution of trigger</td>
<td>105</td>
<td>drop index</td>
</tr>
<tr>
<td>40</td>
<td>grant</td>
<td>106</td>
<td>sp_encryption (encrypted column administration)</td>
</tr>
<tr>
<td>41</td>
<td>insert table</td>
<td>107</td>
<td>create encryption key</td>
</tr>
<tr>
<td>42</td>
<td>insert view</td>
<td>108</td>
<td>Alter Encryption Key as/not default</td>
</tr>
<tr>
<td>43</td>
<td>load database</td>
<td>109</td>
<td>drop encryption key</td>
</tr>
<tr>
<td>44</td>
<td>load transaction</td>
<td>110</td>
<td>deploy user-defined web services</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111</td>
<td>undeploy user defined web services</td>
</tr>
<tr>
<td>45</td>
<td>login</td>
<td>112</td>
<td>login has been locked</td>
</tr>
<tr>
<td>46</td>
<td>logout</td>
<td>113</td>
<td>quiesce hold security</td>
</tr>
<tr>
<td>47</td>
<td>revoke</td>
<td>114</td>
<td>quiesce release</td>
</tr>
<tr>
<td>48</td>
<td>rpc in</td>
<td>115</td>
<td>Password administration</td>
</tr>
<tr>
<td>49</td>
<td>rpc out</td>
<td>116</td>
<td>create manifest file</td>
</tr>
<tr>
<td>50</td>
<td>server boot</td>
<td>117</td>
<td>regenerate keypair</td>
</tr>
<tr>
<td>51</td>
<td>server shutdown</td>
<td>118</td>
<td>alter encryption key modify encryption</td>
</tr>
<tr>
<td>52</td>
<td>Reserved</td>
<td>119</td>
<td>alter encryption key add encryption</td>
</tr>
<tr>
<td>53</td>
<td>Reserved</td>
<td>120</td>
<td>alter encryption key drop encryption</td>
</tr>
<tr>
<td>54</td>
<td>Reserved</td>
<td>121</td>
<td>alter encryption key modify owner</td>
</tr>
<tr>
<td>55</td>
<td>role toggling</td>
<td>122</td>
<td>alter encryption key for key recovery</td>
</tr>
<tr>
<td>56</td>
<td>Reserved</td>
<td>123</td>
<td>LDAP state changes</td>
</tr>
<tr>
<td>57</td>
<td>Reserved</td>
<td>124</td>
<td>alter...modify owner</td>
</tr>
<tr>
<td>58</td>
<td>Reserved</td>
<td>127</td>
<td>Errorlog administration</td>
</tr>
<tr>
<td>59</td>
<td>Reserved</td>
<td>136</td>
<td>transfer table</td>
</tr>
<tr>
<td>60</td>
<td>Reserved</td>
<td>136</td>
<td>transfer table</td>
</tr>
<tr>
<td>61</td>
<td>access to audit table</td>
<td>137</td>
<td>create login profile</td>
</tr>
<tr>
<td>62</td>
<td>select from table</td>
<td>138</td>
<td>alter login</td>
</tr>
<tr>
<td>63</td>
<td>select from view</td>
<td>139</td>
<td>drop login</td>
</tr>
<tr>
<td>64</td>
<td>truncate table</td>
<td>140</td>
<td>alter login profile</td>
</tr>
<tr>
<td>65</td>
<td>Reserved</td>
<td>141</td>
<td>drop login profile</td>
</tr>
<tr>
<td>66</td>
<td>Reserved</td>
<td>142</td>
<td>Reserved</td>
</tr>
<tr>
<td>67</td>
<td>unbind default</td>
<td>143</td>
<td>alter thread pool</td>
</tr>
</tbody>
</table>
Monitoring failed login attempts

The audit option login_locked and the event Locked Login (value 112) record when a login account is locked due to exceeding the configured number of failed login attempts. This event is enabled when audit option login_locked is set. To set login_locked, enter:

```sql
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.

Auditing login failures

Although client applications may fail to login for many reasons, Adaptive Server does not provide them with any detailed information about the login failure. This is done to avoid giving information to malintentioned users attempting to crack passwords or otherwise breach Adaptive Server’s authentication mechanisms.

However, as a system administrator, detailed information is useful for diagnosing Adaptive Server administrative or configuration problems, and it is useful to security officers for investigating attempts to breach security.

This enables auditing for all login failures:

```sql
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.

Adaptive Server audits login failures for the following conditions:

- For Adaptive Server started as a Windows Service, if the Sybase 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 Adaptive Server for Windows with the Trusted Login or Unified Login configuration, but the specified user is not a trusted administrator (that is, an authentication failure).

Adaptive Server does not support the SQL interface requested by the client.

A user is attempting to log into Adaptive Server when it is in single-user mode. In single-user mode, exactly one user with the sa_role is allowed to log in to Adaptive Server. 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 Adaptive Server 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.

Adaptive Server 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.

Adaptive Server’s 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.
Understanding the audit tables

- A shutdown is in progress, but the specified user does not have the sa role.
- Adaptive Server 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.
- Adaptive Server fails to setup tempdb for a login.
- TDS Login Protocol errors are detected.
Symbols

& (ampersand) translated to underscore in login names 129
' (apostrophe) converted to underscore in login names 129
* (asterisk) converted to pound sign in login names 129
select and 208
\ (backslash) translated to underscore in login names 129
^ (caret) converted to dollar sign in login names 129
: (colon) converted to underscore in login names 129
, (comma) converted to underscore in login names 129
{} (curly braces) converted to dollar sign in login names 129
= (equals sign) converted to underscore in login names 129
! (exclamation point) converted to dollar sign in login names 129
< (left angle bracket) converted to dollar sign in login names 129
' (left quote), converted to underscore in login names 129
- (minus sign) converted to pound sign in login names 129
() (parentheses) converted to dollar sign in login names 129
% (percent sign) translated to underscore in login names 129
. (period) converted to dollar sign in login names 129
| (pipe) converted to pound sign in login names 129
+ (plus) converted to pound sign in login names 129

? (question mark) converted to dollar sign in login names 129
“ ” (quotation marks) converted to pound sign in login names 129
enclosing punctuation 17
enclosing values 21
> (right angle bracket) converted to underscore in login names 129
' (right quote), converted to underscore in login names 129
; (semicolon) converted to pound sign in login names 129
/ (slash) converted to pound sign in login names 129
[ ] (square brackets) converted to pound sign in login names 129
~ (tilde) converted to underscore in login names 129
$ISA 170

A
access 221
restricting guest users 69
access control, row level 221
access permissions. See object access permissions
access protection. See permissions; security functions
access rules
alter table command 227
bcp 227
creating 224
creating and binding 222
dropping 223
examples 225
extended 224
sample table 222
accounting, chargeback 89
accounts, server
See logins;users
Index

ACF (Application Context Facility), problem-solving with 237
activating roles 103
Adaptive Server principal name 142
adding
  comments to the audit trail 359
group to a database 71
guest users 69
logins to server 17
remote users 71
users to a group 68
administering security, getting started 5–8
aliases, user
  See also logins; users
  creating 73
database ownership transfer and 180
dropping 74, 75
help on 75
alter role command 56, 59, 98
alternate identity. See alias, user
and (&) 
  translated to underscore in login names 129
ansi_permissions option, set
  permissions and 185
apostrophe converted to underscore in login names 129
Application Context Facility 230, 231
  granting and revoking privileges 232
  valid users 232
application contexts
  built-in functions 233
  using 233
applications
  proxy authorization and 198
assigning
  login names 7
asymmetric key pairs, generating 39
asterisk (*)
  converted to pound sign in login names 129
select and 208
audit options
  displaying 359
  examples 384
  setting 383
audit queue 355, 375
audit queue size configuration parameter 355, 375
audit trail 351, 395
  adding comments 359, 393
  changing current audit table 371
  illustration with multiple audit tables 353
  managing 370
  querying 394
  threshold procedure for 370
auditing 13, 351, 351–394
  See also audit options
  adding comments to the audit trail 359
  configuration parameters 355
  devices for 361
  disabling 359
  displaying options for 359
  enabling 359
  enabling and disabling 378
  installing 361
  installing using the auditinit utility 361
  installing using the installsecurity script 360
  managing the audit trail 370
  managing the transaction log 377
  overview 351
  queue, size of 355
  sybsecurity database 353
  sysaudits_01...sysaudits_08 tables 395
  system procedures for 359
  threshold procedure for 370
  turning on and off 378
auditing configuration parameter 378
authentication 118
  mutual 119
authorizations. See permissions
automatic LDAP enhancements 166
automatic operations
  character conversions in logins 128
  automatic user authentication enhancements 166

B
backslash (\) 
  translated to underscore in login names 129
base tables. See tables
bcp (bulk copy utility)
  security services and 133
  with access rules 227

412
Adaptive Server Enterprise
Index

| built-in functions security | 135 |
| CA certificates 329 | location of 332 | trusted root certificate 329 |
| certificates administration of 337 | authorizing 336 | CA certificates 329 | defined 329 | obtaining 334 | public-key cryptography 329 | requesting 336 | self-signed CA 336 | server certificates 329 |
| chains, ownership 210 | changing database owners 180 | passwords for login accounts 80 | user information 79 | user’s group 72 | user’s identity 194 |
| character sets and password-protected dumps 349 | characters disallowed in login names 128 | chargeback accounting 89 | checking passwords for at least one character 25 |
| cipher suites defined 340 | supported 340 | clients assigning client name, host name, and application name 82 |
| colon (:) converted to underscore in login names 129 | comma (,) converted to underscore in login names 129 | comments adding to audit trail 359, 393 |
| common name, specifying with SSL 347 | concrete identification 185 | concurrent Kerberos authentication 147 |
| configuration (server) network-based security 121 |
| configuration parameters audit-related 355 chargeback accounting 90 |
| configuring Kerberos 137 conflicting permissions 192 See also permissions context-sensitive protection 208 |
| cpu accounting flush interval configuration parameter 90 |
| CPU usage per user 89 |
| create database command permission to use 179 | create login command 17 |
| create rule command, new functionality 221 |
| create rule syntax 221 |
| create rule, syntax 222 |
| creating databases 179 groups 71 guest users 69 login profiles 61 logins 17 subsecurity database 367 user aliases 73 credential delegation 119 credential, security mechanism and 118 curly braces ({}) converted to dollar sign in login names 129 |
| current audit table configuration parameter 371 current usage statistics 89 |
| current user set proxy and 197 custom password checks 33 custom password complexity checks 28 |

D
DAC. See discretionary access control (DAC) data See also permissions integrity of 130
Index

database dumps
  password-protected 348

database object owners
  permissions 179, 195
  status not transferable 83

database objects
  access permissions for 184
  creating 182
  dependent 211
  dropping 182
  dropping users who own 83
  ownership 83, 182
  permissions on 182
  triggers on 214

database owners
  changing 180
  name inside database 74, 83
  objects not transferred between 83
  password forgotten by 94
  permissions of 178, 180
  See also database object owners 177
  setuser command and 194
  several users as same 73

permissions
  databases
    auditing 367
    creation permission 179
    dropping users from 83
    ownership of 179
  database-specific dbcc, master and 188
  dbcc and storage_admin_role command 188
  dbcc (database consistency checker)
    database-specific commands 187, 188
    defined 187
    described 187
    discretionary access control 187
    grant dbcc and roles 188
    grant dbcc and users in databases 188
    grant dbcc checkstorage command and 188
    server-wide commands 188
    tune command and 188
    deactivating roles 103
    denying access to a user 55
    devices
      audit system 361
digital signature
  defined 328
  nonrepudiation 328
  public-key cryptography 328
  tamper detection 328
directory drivers 123
  example of entry in libtcl.cfg file 125
directory entries, creating 337
directory services in libtcl.cfg file 123
  disabling auditing 359
disallowing simple passwords 28
discretionary access control (DAC) 177–214
  See also permissions
  granting and revoking permissions of dbcc commands 187
  overview 11
  stored procedures and 209
  system administrators and 178
  user alias and 194
  views for 207
  drop role command 103
  dropping
    groups 84
    guest users of master 69
    user aliases 74, 75
    user from a database 83
    user-defined roles 103
    users who own database objects 83
dscp utility for specifying security mechanism 122
dsedt utility for security services 122
dump database syntax 348

e
  enable cis configuration parameter 254
  enabling
    auditing 359
    SSL 333
  encryption
    key exchange 328
    public/private key 328
    public-key cryptography 328
    symmetric key 328
  environment variable
    $ISA 170
  exclamation point (!)
converted to dollar sign in login names 129
expand_down parameter
sp_activeroles 106
expiration interval for passwords 35
expiration of passwords 35
exporting set options 247

F
filter parameter, in sp_addserver 348
finding
user IDs 78
user names 78
users in a database 78
functions
security 135

G
get_appcontext 233, 234
global login triggers 249
grant command 178, 183–193
roles and 108
grant dbcc
roles and 188
users in databases and 188
grant option
sp_helpprotect 204
granting
predicated privileges 254
roles to roles 99
roles with grant role 107
system privileges 300
granting default permissions on system tables 189–191
granular permissions
added roles 290
configuring 274
database owner privileges 288
default roles granted to sa 290
enable 274
granted to system-defined roles 284
lock-out server 295
manage any object permission 283
manage database permissions 280
manage security permissions 276
manage server permissions 278
master.dbo.sysprotects 298
restore default database owner privilege 288
restore default role privilege 288
revoking system defined role 293
roles granted to sa 292
sybsecurity 294
system privileges 274
groups
See also public group
changing 72
conflicting permissions and 192
dropping 84
grant and 186
naming 71
revoke and 187
guest users 183
adding 69
creating 69
permissions 69
sample databases and 70
guidelines, security 6

H
hash
defined 328
message digest 328
hierarchy
permissions. See permissions
roles. See role hierarchies
high availability and passwords 52
housekeeper task
license use monitoring 85

I
I/O
usage statistics 89
i/o accounting flush interval configuration parameter 90
identification and authentication
Index

See also logins
  controls 9
identities
  alternate 73
  proxies and 195
  session authorizations and 195
identity of user. See aliases; logins; users
IDs, user 78, 93
individual accountability 7
information (server)
  changing user 79
  locked logins 56
  logins 78
  permissions 203–208
  user aliases 75
  users, database 75–90
installation, server
  audit system 361
  establishing security after 6–8
  interfaces file 122
is_sec_service_on security function 135
isql utility command
  security services and 133

J
joins
  views and 207

K
-k option 142
kadmin 138
Kerberos 136
  compatibility 136
  configuring 137
  CyberSafe Kerberos libraries 136
  keytab file 138
  licenses 136
  MIT Kerberos libraries 136
  Native libraries 136
Kerberos authentication 142
  concurrent 147
  verifying 145
key exchange
  encryption 328
  public/private key 328
  symmetric key 328
key pairs, generating asymmetric 39

L
LAN Manager security mechanism 126
LDAP
  enhancements 165
  setting failback time interval 167
  state transitions 157
  support 155
  syntax 167
LDAP user authentication 159
  password changes 153
  tighter controls on login mapping 160
  troubleshooting 163
  tuning 159
libtcl.cfg file
  example of 125
  preparing for network-based security 123
  tools for editing 124
license information configuration parameter 85
license use
  error log messages 85
  monitoring 84
linking users. See alias, user
list_appcontext 233, 235
load database syntax 348
locking
  logins 22, 55
log on option
  create database 19, 62
logging
  login mapping 163
  login IDs, number of 87
  login mapping
    tighter controls 160
  login names. See logins
  login process
    authentication 118
  login triggers
    and set options 247
configuring 239
disabling execute privilege 247
displaying 241
dropping and changing 240
executing 241
issues 246
issues and information 246
output 241
restrictions 246
restrictions on 246
syntax for configuring 240
syntax for creating 239
understanding output 241
using 239
using for other applications 241
logins
See also remote logins; users
adding to servers 17
alias 74, 75
assigning names for 7
displaying password information 24
finding 78
identification and authentication 9
information on 78
invalid names 128
locking 22, 55
maximum attempts, changing 23
maximum attempts, setting 22
“sa” 6
unlocking 55
lookup server
secondary 155

M
manage any object permission privilege 283
manage database permissions privilege 280
manage security permissions privilege 276
manage server permissions privilege 278
managing users. See users
mapping, login 163
master database
dropping guest users of 69
granting default permissions on system tables 190
guest user in 69
ownership of 180
revoking default permissions on system tables 190
max native threads for LDAP user authentication 159
max roles enabled per user configuration parameter 97
membership keyword, alter role 99
memory
audit records 375
network-based security and 130
message digest
defined 328
hash 328
messages
confidentiality 119, 129
integrity 119, 130
origin checks 119
protection services for 118
minimum
alphabetic characters in password 28
digits in password 28
number of uppercase letters in password 29
minus sign (-)
converted to pound sign in login names 129
mut_excl_roles system function 106
mutual exclusivity of roles 12, 106

N
names
See also information (server); logins
alias 74, 75, 194
finding user 78
for logins 7
original identity 194
user 67, 78, 182
naming
groups 71
user-defined roles 97
network drivers 123
example of entry in libtcl.cfg file 125
syntax for in libtcl.cfg file 123
network-based security 117–135
adding logins for unified login 130
configuring server for 126
Index

connecting to server 133
getting information about 133, 134
identifying users and servers 126
memory requirements 130
process for administering 120
remote procedure calls 131
security mechanism 126
setting up configuration files 121
using 133
nonrepudiation, digital signature 328
NT LAN Manager security mechanism 126
null passwords 81
number of login IDs 87
number of users 87

O
object access permissions See permissions
objectid.dat file 125
location of 337
old and new password complexity checks 31
operator role
permissions 94
order of commands
grant and revoke statements 183–187
out-of-sequence checks 119
ownership chains 210

P
parentheses ( )
converted to dollar sign in login names 129
password
expiration warnings 30
password changes for LDAP user authentication 153
password complexity
cross-checks 30
custom password checks 33
old and new 31
password complexity checks 27
custom password complexity checks 28
disallowing simple passwords 28
password expiration warnings 30
specifying a minimum number of alphabetic characters 28
specifying a minimum number of digits 28
specifying a minimum number of uppercase characters in password 29
password security 53
generating an assymmetric key pair 39
generating key pairs using sp_passwordpolicy 39
securing login passwords on a network 38
password-protected database dumps 348
passwords 80
backward compatibility 40
changing 80
checking for at least one character 25
choosing 21
choosing secure 21
date of last change 77
displaying information 24
downgrading 43
expiration interval 35
expiration of 35
for roles 35
forgotten 94
high availability and 52
minimum length 26
null 81
protecting 21
protection against guessing 22
roles and 103
rules for 21
sp_password 80
percent sign (%)
translated to underscore in login names 129
period (.)
converted to dollar sign in login names 129
permissions
See also discretionary access control (DAC)
acquiring other users’ 193
aliases and 73
ansi_permissions option and 185
concrete identification 185
create database 179
database owners 178, 180
granting 183–193
groups and 71
guest users 69
hierarchy of user 184
Index

information on 203–??
object 182
object access 183, 184–187
operator 94
ownership chains and 210
public group 182, 193
revoking 183–193
selective assignment of 191
stored procedures 182
system administrator 178–179
system procedures 183
system tables 189
tables 182
tables compared to views 207
transfers and 180
triggers and 214
using setuser 194
views 207–208
on views instead of columns 208
Pluggable Authentication Module (PAM) 168
32- and 64-bit servers on the same machine 170
configuring Adaptive Server for PAM 170
determining which module to use 169
enable pam user auth 170
password management 171
unified logins 169
plus (+) 129
converted to pound sign in login names
predicated 259
role activation 259
predicated privileges 252
commands used for configuring 253
granting 254
recompiling 269
revoking 257
SQL behavior 259
preferences, user name 67
principal name 142
for Adaptive Server 142
using sybmapname 143
with SYBASE_PRINCIPAL 142
with the -k option 142
privilege
manage any object permission 283
manage database permissions 280
manage security permissions 276
manage server permissions 278
proc_role system function 106, 209
stored procedures and reports 203–??
processes (server tasks)
See also servers
administering Adaptive Server 5
current on server 76
information on 76
protection mechanisms. See security functions; stored procedures; views
protection system
context-sensitive 208
hierarchy (ownership chains) 210
reports 203–??
proxy authorization 193–??
executing 196
how applications use it 198
how users use it 196
overview 195
using 195, 197
public group 71
See also groups
guest user permissions and permissions 182, 193
sp_adduser and 68
sp_changegroup and 72
public/private key encryption 328
public-key cryptography
certificates 328
defined 328
digital signature 328
encryption 328
Q
quotation marks (" ")
converted to pound sign in login names 129

Security Administration Guide 419
Index

R

records, audit 355
reestablishing original identity 194
remote procedure calls
   network-based security 131
   unified login and 132
remote server users. See remote logins
remote users. See remote logins
replay detection 119
reporting
   usage statistics 89
reports
   server usage 89
revoke command 178, 183–193
revoking
   default permissions from system tables 190
   roles with revoke role 109
revoking default permissions on master database system tables 190
RFC 86.0 169
rm_appcontext 233, 236
role hierarchies 12
   creating 108
   displaying 106
   displaying with role_contain 105
   displaying with sp_displayroles 105
role_contain system function 105
roles
   activating 103
   configured for sa login 6
   deactivating 103
   locking 22, 55
   maximum login attempts, changing 23
   maximum login attempts, setting 22
   passwords for 35
   permissions and 108, 184
   stored procedure permissions and 106
   stored procedures and 108, 209
   unlocking 55, 56, 59
roles, user-defined
   planning 96
row level access control 221
rules
   protection hierarchy 213

S

“sa” login 6
   changing password for 7
   configured with system administrator and system security officer roles 6
   security recommendations for using 6
secmech specification 125
secondary
   lookup server support 155
   lookup servers, using sp_ldapadmin 156
secure default login 127
securing login passwords on a network 38
security
   auditing 13
   discretionary access control 11
   establishing after installation 6–8
   identification and authentication controls 9
   Kerberos 136
   roles 11
security administration
   example of 7
   getting started 5–8
   guidelines 6
security drivers
   example of entry in libtcl.cfg file 125
   syntax for entries in libtcl.cfg file 123
security functions 135
security mechanisms 134
security services
   example 118–119
   supported by Adaptive Server 119
select * command
   error message 208
sensitive information, views of 207
separation of roles 11
sequence checks 119
server authentication
   server certificates 331
server certificates 329
   location of 332
server authentication 331
server user name and ID 78
servers
   adding new logins to 17
   adding users to 17
   user information 75–90
Index

server-wide **dbcc, master** and 188
**session authorization** option, **set** 197
**set** command
roles and 103
set options
exportable 247
**set appcontext** 233
setting timeout for LDAP user authentication 159
**setuser** command
**show_role** and 105
**setuser**, using 194
**show role** system function 105
**show sec services** security function 135
slash (/)
converted to pound sign in login names 129
**sp.activeroles** system procedure 106
**sp.addalias** system procedure 74
**sp.addauditrecord** system procedure 393
**sp.addgroup** system procedure 71
**sp.addlogin** system procedure 35, 37
**sp.addserver**
includes filter parameter 348
**sp.adduser** system procedure 69
**sp.audit** system procedure
setting options with 383
**sp.changedbowner** system procedure 180
**sp.changegroup** system procedure 71, 72
**sp.configure** system procedure
configuring server for security services 126
**sp.displaylogin** system procedure 77
**sp.displayroles** system procedure 105
**sp.dropalias** system procedure 74, 75
**sp.dropgroup** system procedure 84
**sp.dropuser** system procedure 83
**sp.help** system procedure 204–205
**sp.helprotect** system procedure 75
**sp.listener**, specifying a common name 347
**sp.locklogin** system procedure 56
sp_logintrigger 249
sp_maplogin 160
**sp.modifylogin** system procedure 35, 38
**sp.password** system procedure 80
**sp.passwordpolicy** syntax 39
**sp.reportstats** system procedure 89
sp_serveroption net password encryption description 40
**sp.who** system procedure 76, 204
square brackets []
converted to pound sign in login names 129
**SSL**
common name, to specify 347
defined 330
enabling SSL 333
filter, defined 331
handshake 330
SSL connections
for companion servers 333
for RPCs 333
Open Client 333
state transitions
LDAP server 157
statistics
I/O usage 89
steps
administering security 5
stored procedures
checking for roles in 106
granting execution permission to roles 106
ownership chains 210
permissions on 182
roles and 209
as security mechanisms 209
**suser_id** system function 78–79
**suser_name** system function 78–79
**suspend audit when device full** configuration parameter 376
syb_map_name 143
SYBASE_PRINCIPAL 142
syblicenseslog table 86
sybmapname 143
sybsecurity database 353
sybsystemprocs database
permissions and 183
symmetric key encryption 328
syntax
dump database 348
load database 348
**sys_session** application context table 237
sysalternates table 74
See also sysusers table
Index

syslogs transaction log for sybsecurity 377
sys.servers table
  sp_helpserver and 133
system administrator
  permissions 178–179
system audit tables 395
system procedures
  for changing user information 79–83
  for dropping aliases 75
  permissions 183
system roles
  activating 103
  deactivating 103
  granting with grant role 107
max_roles_enabled configuration parameter and 97
show_role and 105
system tables
  changes allowed to 189
  permissions on 189
sysusers table
  permissions and 183
  sysalternates table and 74

T
  tables
    context-sensitive protection of 208
    ownership chains for 210
    permissions on 182
    permissions on, compared to views 207
  underlying 207
  tamper detection, digital signature 328
  threshold procedures
    audit trail 370
triggers
  permissions and 214
  troubleshooting LDAP user authentication 163
trusted root certificate
  CA certificate 329
  location of 332
  tuning
    LDAP user authentication 159

U
  underlying tables of views (base tables) 207
  unified login 119
    mapping login names 128
  remote procedure security models 132
  requiring 127
  secure default login 127
unlocking
  login accounts 55
  roles 56, 59
updating
  system procedures and 209
usage
  statistics 89
use security services configuration parameter 127
  user authentication enhancements 165
user databases
  See also databases; permissions
user groups. See groups; public group
user IDs 93
  displaying 77
  finding 78
user names 78, 182
  finding 78
  preferences 67
user_id system function 79
user_name system function 79
user-defined roles
  activating 103
  deactivating 103
  dropping 103
  granting with grant role 107
  number of 97
  planning 96
users
  See also aliases; groups; logins; remote logins
  adding 67, 71
  aliases 73
  application name, setting 82
  client host name, setting 82
  client name, setting 82
  currently on database 76
  currently on server 76
  dropping from databases 83–84
  dropping from groups 72
  guest 69, 183
IDs 78, 93
information on 75–90
license use monitoring 84
number or 87
permissions to all or specific 191, 208
views for specific 207
visiting 70
using proxy authorization 195

V
verifying Kerberos authentication 145
views
dependent 211
ownership chains 210
permissions on 207–208
security and 206
visitor accounts 70

W
Windows NT LAN Manager security mechanism 126