SYBASE®

System Administration Guide

Adaptive Server® Enterprise

12.5.1

DOCUMENT ID: DC31654-01-1251-02

LAST REVISED: November 2004

Contents

About This Book

This manual, the *Sybase Adaptive Server System Administration Guide,* describes how to administer and control Sybase® Adaptive Server® Enterprise databases independent of any specific database application.

Audience

This manual is for Sybase System Administrators and Database Owners.

How to use this book

This manual contains the following chapters:

- Chapter 1, “Overview of System Administration,” describes the structure of the Sybase system.
- Chapter 2, “System and Optional Databases,” discusses the contents and function of the Adaptive Server system databases.
- Chapter 3, “System Administration for Beginners,” summarizes important tasks that new System Administrators need to perform.
- Chapter 4, “Setting Configuration Parameters,” summarizes the configuration parameters that you set with sp_configure, which control many aspects of Adaptive Server behavior.
- Chapter 5, “Diagnosing System Problems,” discusses Adaptive Server and Backup Server™ error handling and shows how to shut down servers and kill user processes.
- Chapter 6, “Limiting Access to Server Resources,” explains how to create and manage resource limits with Adaptive Server.
- Chapter 7, “Configuring Character Sets, Sort Orders, and Languages,” discusses international issues, such as the files included in the Language Modules and how to configure an Adaptive Server language, sort order, and character set.
- Chapter 8, “Configuring Client/Server Character Set Conversions,” discusses character set conversion between Adaptive Server and clients in a heterogeneous environment.
- Chapter 9, “Security Administration,” provides an overview of the security features available in Adaptive Server.

- Chapter 10, “Managing Adaptive Server Logins, Database Users, and Client Connections,” describes methods for managing Adaptive Server login accounts and database users.
- Chapter 11, “Auditing,” describes how to set up auditing for your installation.
- Chapter 12, “Managing User Permissions,” describes the use and implementation of user permissions.
- Chapter 13, “Managing Remote Servers,” discusses the steps the System Administrator and System Security Officer of each Adaptive Server must execute to enable remote procedure calls (RPCs).
- Chapter 14, “Using Kerberos, DCE, and Windows NT LAN Manager,” describes the network-based security services that enable you to authenticate users and protect data transmitted among machines on a network.
- Chapter 15, “Overview of Disk Resource Issues,” provides an overview of Adaptive Server disk resource issues.
- Chapter 16, “Initializing Database Devices,” describes how to initialize and use database devices.
- Chapter 17, “Mirroring Database Devices,” describes how to mirror database devices for nonstop recovery from media failures.
- Chapter 18, “Configuring Memory,” explains how to configure Adaptive Server to use the available memory on your system.
- Chapter 19, “Configuring Data Caches,” discusses how to create named caches in memory and bind objects to those caches.
- Chapter 20, “Managing Multiprocessor Servers,” explains how to use multiple CPUs with Adaptive Server and discusses system administration issues that are unique to symmetric multiprocessing (SMP) environments.
- Chapter 21, “Creating and Managing User Databases,” discusses the physical placement of databases, tables, and indexes, and the allocation of space to them.
- Chapter 22, “Database Mount and Unmount,” describes how to transport databases from a source Adaptive Server to a destination Adaptive Server.
- Chapter 23, “Setting Database Options,” describes how to set database options.

- Chapter 24, “Creating and Using Segments,” describes how to use segments, which are named collections of database devices, in databases.
- Chapter 25, “Using the reorg Command,” describes how to use the reorg command.
- Chapter 26, “Checking Database Consistency,” describes how to use the database consistency checker, dbcc, to detect and fix database problems.
- Chapter 27, “Developing a Backup and Recovery Plan,” discusses the capabilities of the Backup Server and how to develop your backup strategy.
- Chapter 28, “Backing Up and Restoring User Databases,” discusses how to recover user databases.
- Chapter 29, “Restoring the System Databases,” discusses how to recover system databases.
- Chapter 30, “Automatic Database Expansion,” describes how to configure databases to expand automatically when they run out of space.
- Chapter 31, “Managing Free Space with Thresholds,” discusses managing space with thresholds.

Related documents

The Sybase Adaptive Server Enterprise documentation set consists of the following:

- The release bulletin for your platform – contains last-minute information that was too late to be included in the books.

  A more recent version of the release bulletin may be available on the World Wide Web. To check for critical product or document information that was added after the release of the product CD, use the Sybase Technical Library.
- The *Installation Guide* for your platform – describes installation, upgrade, and configuration procedures for all Adaptive Server and related Sybase products.
- *What’s New in Adaptive Server Enterprise?* – describes the new features in Adaptive Server version 12.5.1, the system changes added to support those features, and the changes that may affect your existing applications.
- *ASE Replicator User’s Guide* – describes how to use the ASE Replicator feature of Adaptive Server to implement basic replication from a primary server to one or more remote Adaptive Servers.

- *Component Integration Services User’s Guide* – explains how to use the Adaptive Server Component Integration Services feature to connect remote Sybase and non-Sybase databases.
- *Configuring Adaptive Server Enterprise* for your platform – provides instructions for performing specific configuration tasks for Adaptive Server.
- *EJB Server User’s Guide* – explains how to use EJB Server to deploy and execute Enterprise JavaBeans in Adaptive Server.
- *Error Messages and Troubleshooting Guide* – explains how to resolve frequently occurring error messages and describes solutions to system problems frequently encountered by users.
- *Full-Text Search Specialty Data Store User’s Guide* – describes how to use the Full-Text Search feature with Verity to search Adaptive Server Enterprise data.
- *Glossary* – defines technical terms used in the Adaptive Server documentation.
- *Historical Server User’s Guide* – describes how to use Historical Server to obtain performance information for SQL Server® and Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as data types, functions, and stored procedures in the Adaptive Server database.
- *Job Scheduler User's Guide* – provides instructions on how to install and configure, and create and schedule jobs on a local or remote Adaptive Server using the command line or a graphical user interface (GUI).
- *Monitor Client Library Programmer’s Guide* – describes how to write Monitor Client Library applications that access Adaptive Server performance data.
- *Monitor Server User’s Guide* – describes how to use Monitor Server to obtain performance statistics from SQL Server and Adaptive Server.
- *Performance and Tuning Guide* – is a series of four books that explains how to tune Adaptive Server for maximum performance:
    - *Basics* – the basics for understanding and investigating performance questions in Adaptive Server.
    - *Locking* – describes how the various locking schemas can be used for improving performance in Adaptive Server.

    - *Optimizer and Abstract Plans* – describes how the optimizer processes queries and how abstract plans can be used to change some of the optimizer plans.
    - *Monitoring and Analyzing* – explains how statistics are obtained and used for monitoring and optimizing performance.
- *Quick Reference Guide* – provides a comprehensive listing of the names and syntax for commands, functions, system procedures, extended system procedures, datatypes, and utilities in a pocket-sized book.
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL® information:
    - *Building Blocks* – Transact-SQL datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
    - *Commands* – Transact-SQL commands.
    - *Procedures* – Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
    - *Tables* – Transact-SQL system tables and dbcc tables.
- *System Tables Diagram* – illustrates system tables and their entity relationships in a poster format. Available only in print version.
- *Transact-SQL User's Guide* – documents Transact-SQL, Sybase's enhanced version of the relational database language. This manual serves as a textbook for beginning users of the database management system. This manual also contains descriptions of the pubs2 and pubs3 sample databases.
- *Using Adaptive Server Distributed Transaction Management Features* – explains how to configure, use, and troubleshoot Adaptive Server DTM features in distributed transaction processing environments.
- *Using Sybase Failover in a High Availability System* – provides instructions for using Sybase's Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Utility Guide* – documents the Adaptive Server utility programs, such as isql and bcp, which are executed at the operating system level.
- *Web Services User's Guide* – explains how to configure, use, and troubleshoot Web Services for Adaptive Server.

- *XA Interface Integration Guide for CICS, Encina, and TUXEDO* – provides instructions for using the Sybase DTM XA interface with X/Open XA transaction managers.
- *XML Services in Adaptive Server Enterprise* – describes the Sybase native XML processor and the Sybase Java-based XML support, introduces XML in the database, and documents the query and mapping functions that comprise XML Services.

Other sources of information

Use the Sybase Getting Started CD, the Sybase Technical Library CD and the Technical Library Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the Technical Library CD. It is included with your software. To read or print documents on the Getting Started CD you need Adobe Acrobat Reader (downloadable at no charge from the Adobe Web site, using a link provided on the CD).
- The Technical Library CD contains product manuals and is included with your software. The DynaText reader (included on the Technical Library CD) allows you to access technical information about your product in an easy-to-use format.

  Refer to the *Technical Library Installation Guide* in your documentation package for instructions on installing and starting the Technical Library.
- The Technical Library Product Manuals Web site is an HTML version of the Technical Library CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Updates, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Technical Library Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

v **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.

5 Click a Certification Report title to display the report.

v **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1 Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.

2 Click MySybase and create a MySybase profile.

Sybase EBFs and software updates

v **Finding the latest information on EBFs and software updates**

1 Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.

2 Select EBFs/Updates. Enter user name and password information, if prompted (for existing Web accounts) or create a new account (a free service).

3 Select a product.

4 Specify a time frame and click Go.

5 Click the Info icon to display the EBF/Update report, or click the product description to download the software.

Conventions

This section describes the style conventions used in this manual.

Formatting SQL statements

SQL is a free-form language: there are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented.

SQL syntax conventions

Table 1 lists the conventions for syntax statements in this manual:

Table 1: Syntax statement conventions

Key	Definition
`command`	Command names, command option names, utility names, utility flags, and other keywords are in `bold Courier` in syntax statements, and in **bold Helvetica** in paragraph text.
`variable`	Variables, or words that stand for values that you fill in, are in italics.
`{ }`	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
`[ ]`	Square brackets mean choosing one or more of the enclosed options is optional. Do not include brackets in your option.
`( )`	Type parentheses as part of the command.
|	The vertical bar means you may select only one of the options shown.
`,`	The comma means you may choose as many of the options shown as you like, separating your choices with commas.

- Syntax statements (displaying the syntax and all options for a command) are printed like this:

  ```
  sp_dropdevice [device_name]
  ```

  or, for a command with more options:

  ```
  select column_name
      from table_name
      where search_conditions
  ```

  In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase: normal font for keywords, italics for user-supplied words.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```

- Examples of output from the computer are printed like this:

```
pub_id  pub_name                  city        state
```

```
------- ------------------      ----------- -----
0736    New Age Books            Boston      MA
0877    Binnet & Hardley         Washington  DC
1389    Algodata Infosystems     Berkeley    CA

(3 rows affected)
```

Case

You can disregard case when you type keywords:

SELECT is the same as Select is the same as select.

Obligatory options {you must choose at least one}

- *Curly braces and vertical bars:* Choose *one and only one* option.

```
{die_on_your_feet | live_on_your_knees | live_on_your_feet}
```

- *Curly braces and commas:* Choose *one or more* options. If you choose more than one, separate your choices with commas.

```
{cash, check, credit}
```

Optional options

- *One item in square brackets:* You don't have to choose it.

```
[anchovies]
```

- *Square brackets and vertical bars:* Choose *none or only one*.

```
[beans | rice | sweet_potatoes]
```

- *Square brackets and commas:* Choose *none, one, or more than one* option. If you choose more than one, separate your choices with commas.

```
[extra_cheese, avocados, sour_cream]
```

Ellipsis

An ellipsis (. . .) means that you can *repeat* the last unit as many times as you like. In this syntax statement, buy is a required keyword:

```
buy thing = price [cash | check | credit]
    [, thing = price [cash | check | credit]]...
```

You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

An ellipsis can also be used inline to signify portions of a command that are left out of a text example. The following syntax statement represents the complete create database command, even though required keywords and other options are missing:

```
create database...for load
```

Expressions

Several different types of **expressions** are used in Adaptive Server syntax statements.

Table 2: Types of expressions used in syntax statements

Usage	Definition
expression	Can include constants, literals, functions, column identifiers, variables, or parameters
logical_expression	An expression that returns TRUE, FALSE, or UNKNOWN
constant_expression	An expression that always returns the same value, such as “5+3” or “ABCDE”
float_expr	Any floating-point expression or expression that implicitly converts to a floating value
integer_expr	Any integer expression or an expression that implicitly converts to an integer value
numeric_expr	Any numeric expression that returns a single value
char_expr	An expression that returns a single character-type value
binary_expression	An expression that returns a single binary or varbinary value

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Overview of System Administration

This chapter introduces the basic topics of Adaptive Server system administration.

Adaptive Server administration tasks

Administering Adaptive Server includes tasks such as:

- Installing Adaptive Server and Backup Server
- Creating and managing Adaptive Server login accounts
- Granting roles and permissions to Adaptive Server users
- Managing and monitoring the use of disk space, memory, and connections
- Backing up and restoring databases
- Diagnosing system problems
- Configuring Adaptive Server to achieve the best performance

In addition, System Administrators may have a hand in certain database design tasks, such as enforcing integrity standards. This function may overlap with the work of application designers.

Although a System Administrator concentrates on tasks that are independent of the applications running on Adaptive Server, he or she is likely to be the person with the best overview of all the applications. For this reason, a System Administrator can advise application designers about the data that already exists on Adaptive Server, make recommendations about standardizing data definitions across applications, and so on.

However, the distinction between what is specific to an application is sometimes a bit “fuzzy.” Owners of user databases will consult certain sections of this book. Similarly, System Administrators and Database Owners will use the *Transact-SQL User’s Guide* (especially the chapters on data definition, stored procedures, and triggers). Both System Administrators and application designers will use the *Performance and Tuning Guide*.

Roles required for system administration tasks

Many of the commands and procedures discussed in this manual require the System Administrator or System Security Officer role. Other sections in this manual are relevant to Database Owners. A Database Owner’s user name within the database is “dbo”. You cannot log in as “dbo:” a Database Owner logs in under his or her Adaptive Server login name and is recognized as “dbo” by Adaptive Server only while he or she is using the database.

Various security-related, administrative, and operational tasks are grouped into the following system roles:

- **System Administrator** – by default the system administrator (the sa) has the following roles:
  - sa_role
  - sso_role
  - oper_role
  - sybase_ts_role

  The system administrator’s tasks include:
  - Managing disk storage
  - Monitoring Adaptive Server’s automatic recovery procedure
  - Fine-tuning Adaptive Server by changing configurable system parameters
  - Diagnosing and reporting system problems

    - Backing up and loading databases
    - Modifying and dropping server login accounts
    - Granting and revoking the System Administrator role
    - Granting permissions to Adaptive Server users
    - Creating user databases and granting ownership of them
    - Setting up groups which can be used for granting and revoking permissions)
- **System Security Officer** – who performs security-related tasks such as:
    - Creating server login accounts, which includes assigning initial passwords
    - Changing the password of any account
    - Granting and revoking the System Security Officer and Operator roles
    - Creating, granting, and revoking user-defined roles
    - Granting the capability to impersonate another user throughout the server
    - Setting the password expiration interval
    - Setting up Adaptive Server to use network-based security services
    - Managing the audit system
- **Operator** – a user who can back up and load databases on a server-wide basis. The operator role allows a single user to use the dump database, dump transaction, load database, and load transaction commands to back up and restore all databases on a server without having to be the owner of each one. These operations can be performed in a single database by the Database Owner or a System Administrator.

These roles provide individual accountability for users performing operational and administrative tasks. Their actions can be audited and attributed to them. A System Administrator operates outside the discretionary access control (DAC) protection system; that is, when a System Administrator accesses objects Adaptive Server does not check the DAC permissions.

In addition, two kinds of object owners have special status because of the objects they own. These ownership types are:

- Database Owner
- Database object owner

Database Owner

The **Database Owner** is the creator of a database or someone to whom database ownership has been transferred. A System Administrator grants users the authority to create databases with the grant command.

A Database Owner logs in to Adaptive Server using his or her assigned login name and password. In other databases, that owner is known by his or her regular user name. In the database Adaptive Server recognizes the user as having the "dbo" account.

A Database Owner can:

- Run the system procedure sp_adduser to allow other Adaptive Server users access to the database
- Use the grant command to give other users permission to create objects and execute commands within the database

Adding users to databases is discussed in Chapter 10, "Managing Adaptive Server Logins, Database Users, and Client Connections." Granting permissions to users is discussed in Chapter 12, "Managing User Permissions."

The Database Owner does not automatically receive permissions on objects owned by other users. However, a Database Owner can temporarily assume the permissions of other users in the database at any time by using the setuser command. Using a combination of the setuser and grant commands, the Database Owner can acquire permissions on any object in the database.

Note Because the Database Owner role is so powerful, the System Administrator should plan carefully who should own databases in the server. The System Security Officer should consider auditing the database activity of all Database Owners.

Database object owner

A **Database object owner** is a user who creates a database object. **Database objects** are tables, indexes, views, defaults, triggers, rules, constraints, and procedures. Before a user can create a database object, the Database Owner must grant the user permission to create objects of a particular type. There is no special login name or password for a database object owner.

The database object owner creates an object using the appropriate create statement, and then grants permission to other users.

The creator of a database object is automatically granted all permissions on that object. The System Administrator also has all permissions on the object. The owner of an object must explicitly grant permissions to other users before they can access the object. Even the Database Owner cannot use an object directly unless the object owner grants him or her the appropriate permission. However, the Database Owner can always use the setuser command to impersonate any other user in the database, including the object owner.

Note When a database object is owned by someone other than the Database Owner, the user (including a System Administrator) must qualify the name of that object with the object owner's name—*ownername.objectname*—to access the object. If an object or a procedure needs to be accessed by a large number of users, particularly in ad hoc queries, having these objects owned by "dbo" greatly simplifies access.

Using *isql* to perform system administration tasks

This book assumes that you will perform the system administration tasks described in this guide by using the command-line utility isql. This section provides some basic information about using isql. For complete information about isql, see the *Utility Guide*.

You can also use the graphic tool Sybase Central™ to perform many of the tasks described in this book, as described in "Using Sybase Central for system administration tasks" on page 6.

Starting *isql*

To start isql on most platforms, type this command at an operating system prompt, where *username* is the user name of the System Administrator:

```
isql -Uusername
```

Adaptive Server prompts you for your password.

Note Do not use the -P option of isql to specify your password because another user might then see your password.

You can use isql in command-line mode to enter many of the Transact-SQL examples in this manual.

Entering statements

The statements that you enter in isql can span several lines. isql does not process statements until you type “go” on a separate line. For example:

```
1> select *
2> from sysobjects
3> where type = "TR"
4> go
```

The examples in this manual do not include the go command between statements. If you are typing the examples, you must enter the go command to see the sample output.

Saving and reusing statements

This manual frequently suggests you that save the Transact-SQL statements you use to create or modify user databases and database objects. The easiest way to do this is to create or copy the statements to an ASCII-formatted file. You can then use the file to supply statements to isql if you need to re-create databases or database objects later.

The syntax for using isql with an ASCII-formatted file is the following, where *filename* is the full path and file name of the file that contains Transact-SQL statements:

```
isql -Uusername -ifilename
```

On UNIX and other platforms, use the less than symbol (<) to redirect the file.

The Transact-SQL statements in the ASCII file must use valid syntax and the go command.

Using Sybase Central for system administration tasks

You can accomplish many of the system administration tasks detailed in this book with Sybase Central, a graphic tool that comes with Adaptive Server.

Here are some of the tasks you can use Sybase Central for:

- Initializing database devices (Windows NT servers only)
- Setting configuration parameters
- Viewing the amount of free log space in a database
- Generating data definition language (DDL)

- Creating logins
- Adding remote servers
- Creating databases
- Creating stored procedures
- Defining roles
- Adding data caches
- Setting database options
- Backing up and restoring databases

You can also use the Monitor Viewer feature of Sybase Central to access Adaptive Server Monitor™. Sybase Central also comes with extensive online help.

You can use the Sybase Central DDL-generation feature to record your work to Transact-SQL scripts. The DDL-generation feature lets you save to a script the actions you performed in an entire server or within a specific database.

System tables

The master database contains **system tables** that keep track of information about Adaptive Server as a whole. In addition, each database (including the master database) contains system tables that keep track of information specific to that database.

All the Adaptive Server-supplied tables in the master database (Adaptive Server's controlling database) are considered system tables. Each user database is created with a subset of these system tables. The system tables may also be referred to as the **data dictionary** or the system catalogs.

A master database and its tables are created when Adaptive Server is installed. The system tables in a user database are created when the create database command is issued. The names of all system tables start with "sys". You cannot create tables in user databases that have the same names as system tables. An explanation of the system tables and their columns is included in the *Reference Manual*.

Querying the system tables

You can query system tables just like any other tables. For example, the following statement returns the names of all the triggers in the database:

```
select name
from sysobjects
where type = "TR"
```

In addition, Adaptive Server supplies **stored procedure***s* (called **system procedures**), many of which provide shortcuts for querying the system tables.

Here are the system procedures that provide information from the system tables:

• sp_commonkey	• sp_helpremotelogin
• sp_configure	• sp_help_resource_limit
• sp_countmedatada	• sp_helprotect
• sp_dboption	• sp_helpsegment
• sp_estspace	• sp_helpserver
• sp_help	• sp_helpsort
• sp_helpartition	• sp_helptext
• sp_helpcache	• sp_helpthreshold
• sp_helpconfig	• sp_helpuser
• sp_helpconstraint	• sp_lock
• sp_helpdb	• sp_monitor
• sp_helpdevice	• sp_monitorconfig
• sp_helpgroup	• sp_showcontrolinfo
• sp_helpindex	• sp_showexeclass
• sp_helpjava	• sp_showplan
• sp_helpjoins	• sp_spaceused
• sp_helpkey	• sp_who
• sp_helplanguage	• sp_help_resource_limit
• sp_helplog	

For complete information about the system procedures, see the *Reference Manual*.

Keys in system tables

Primary, foreign, and common keys for the system tables are defined in the master and model databases. You can get a report on defined keys by executing sp_helpkey. For a report on columns in two system tables that are likely join candidates, execute sp_helpjoins.

The *Adaptive Server System Tables Diagram* included with Adaptive Server shows the relationships between columns in the system tables.

Updating system tables

The Adaptive Server system tables contain information that is critical to the operation of your databases. Under ordinary circumstances, you do not need to perform direct data modifications to system tables.

Update system tables only when you are instructed to do so by Sybase Technical Support or by an instruction in the *Error Messaging and Troubleshooting Guide* or in this manual.

When you update system tables, you must issue an sp_configure command that enables system table updates. While this command is in effect, any user with appropriate permission can modify a system table. Other requirements for direct changes to system tables are:

- Modify system tables only inside a transaction. Issue a begin transaction command before you issue the data modification command.
- Verify see that only the rows you wanted changed were affected by the command and that the data was changed correctly.

- If the command was incorrect, issue a rollback transaction command. If the command was correct, issue a commit transaction command.

Warning! Some system tables should not be altered by any user under any circumstances. Some system tables are built dynamically by system processes, contain encoded information, or display only a portion of their data when queried. Imprudent, ad hoc updates to certain system tables can make Adaptive Server unable to run, make database objects inaccessible, scramble permissions on objects, or terminate a user session.
Moreover, you should never attempt to alter the definition of the system tables in any way. For example, do not alter system tables to include constraints. Triggers, defaults, and rules are not allowed in system tables. If you try to create a trigger or bind a rule or default to a system table, you will get an error message.

System procedures

The names of all system procedures begin with “sp_”. They are located in the sybsystemprocs database, but you can run many of them in any database by issuing the stored procedure from the database or by qualifying the procedure name with the database name.

If you execute a system procedure in a database other than sybsystemprocs, it operates on the system tables in the database from which it was executed. For example, if the Database Owner of pubs2 runs sp_adduser from pubs2 or issues the command pubs2..sp_adduser, the new user is added to pubs2..sysusers. However, this does not apply to system procedures that update only tables in the master database.

Permissions on system procedures are discussed in the *Reference Manual*.

Using system procedures

A **parameter** is an argument to a stored or system procedure. If a parameter value for a system procedure contains reserved words, punctuation, or embedded blanks, it must be enclosed in single or double quotes. If the parameter is an object name, and the object name is qualified by a database name or owner name, the entire name must be enclosed in single or double quotes.

System procedures can be invoked by sessions using either chained or unchained transaction mode. However, the system procedures that modify data in system tables in the master database cannot be executed from within a transaction, since this could compromise recovery. The system procedures that create temporary work tables cannot be run from transactions.

If no transaction is active when you execute a system procedure, Adaptive Server turns off chained mode and sets transaction isolation level 1 for the duration of the procedure. Before returning, the session's chained mode and isolation level are reset to their original settings. For more information about transaction modes and isolation levels, see the *Reference Manual*.

All system procedures report a return status. For example, the following means that the procedure executed successfully:

```
return status = 0
```

System procedure tables

The system procedures use several *system procedure tables* in the master and sybsystemdb databases to convert internal system values (for example, status bits) into human-readable format. One of these tables, spt_values, is used by a variety of system procedures, including:

• sp_configure	• sp_helpdevice
• sp_dboption	• sp_helpindex
• sp_depends	• sp_helpkey
• sp_help	• sp_helprotect
• sp_helpdb	• sp_lock

The spt_values table can be updated only by an upgrade; it cannot be modified otherwise. To see how it is used, execute sp_helptext and look at the text for one of the system procedures that references it.

The other system procedure tables are spt_monitor, spt_committab, and tables needed by the catalog stored procedures. (The spt_committab table is located in the sybsystemdb database.)

In addition, several of the system procedures create and then drop temporary tables. For example, sp_helpdb creates #spdbdesc, sp_helpdevice creates #spdevtab, and sp_helpindex creates #spindtab.

Creating system procedures

Many of the system procedures are explained in this manual, in the sections where they are relevant. For complete information about system procedures, see the *Reference Manual*.

System Administrators can write system procedures that can be executed in any database. Simply create a stored procedure in sybsystemprocs and give it a name that begins with “sp_”. The uid of the stored procedure must be 1, the uid of the Database Owner.

Most of the system procedures that you create query the system tables. You can also create stored procedures that modify the system tables, although this is not recommended.

To create a stored procedure that modifies system tables, a System Security Officer must first turn on the allow updates to system tables configuration parameter. Any stored procedure created while this parameter is set to “on” will *always* be able to update system tables, even when allow updates to system tables is set to “off.” To create a stored procedure that updates the system tables:

1. Use sp_configure to set allow updates to system tables to “on.”
2. Create the stored procedure with the create procedure command.
3. Use sp_configure to set allow updates to system tables to “off.”

Warning! Use extreme caution when you modify system tables. Always test the procedures that modify system tables in development or test databases, not in your production database.

System extended stored procedures

An extended stored procedure (ESP) provides a way to call external language functions from within Adaptive Server. Adaptive Server provides a set of ESPs; users can also create their own. The names of all system extended stored procedures begin with "xp_", and are located in the sybsystemprocs database.

One very useful system ESP is xp_cmdshell, which executes an operating system command on the system that is running Adaptive Server.

You can invoke a system ESP just like a system procedure. The difference is that a system ESP executes procedural language code rather than Transact-SQL statements. All ESPs are implemented by an Open Server application called XP Server, which runs on the same machine as Adaptive Server. XP Server starts automatically on the first ESP innovation.

For information about the system ESPs provided with Adaptive Server, see the *Reference Manual*.

Creating system ESPs

Create a system ESP in the sybsystemprocs database using the create procedure command. System procedures are automatically included in the sybsystemprocs database. The name of the ESP, and its procedural language function, should begin with "xp_". The uid of the stored procedure must be 1, the uid of the Database Owner.

For general information about creating ESPs, see Chapter 16, "Using Extended Stored Procedures," in the *Transact-SQL User's Guide*.

Logging error messages

Adaptive Server writes start-up information to a local error log file each time it starts. The installation program automatically sets the error log location when you configure a new Adaptive Server. See the configuration documentation for your platform to learn the default location and file name of the error log.

Many error messages from Adaptive Server go to the user's terminal only. However, fatal error messages (severity levels 19 and above), kernel error messages, and informational messages from Adaptive Server are recorded in the error log file.

Adaptive Server keeps the error log file open until you stop the server process. If you need to reduce the size of the error log by deleting old messages, stop the Adaptive Server process before you do so.

Note On some platforms such as Windows NT, Adaptive Server also records error messages in the operating system event log. See the Adaptive Server installation and configuration guide for additional information about error logs.

Connecting to Adaptive Server

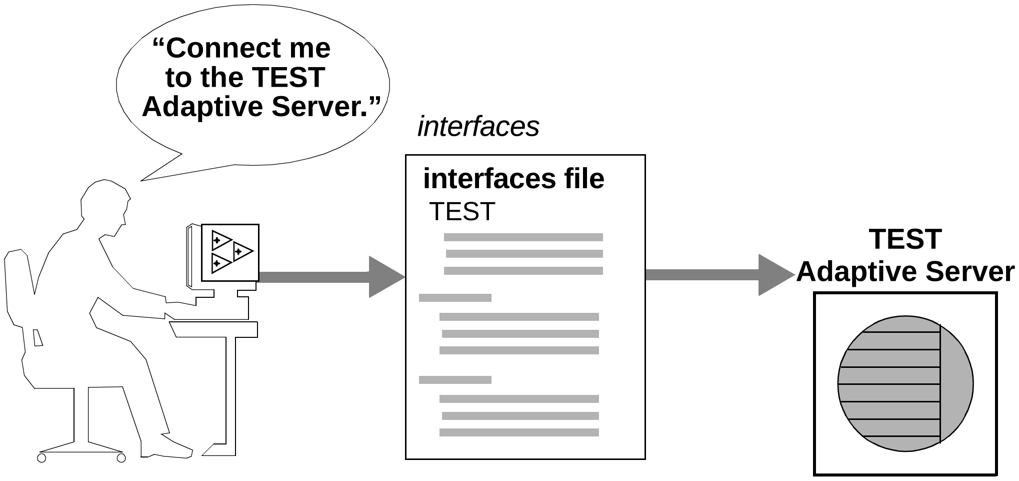
Adaptive Server can communicate with other Adaptive Servers, Open Server applications, and client software on the network. Clients can talk to one or more servers, and servers can communicate with other servers via remote procedure calls. In order for products to interact with one another, each needs to know where the others reside on the network. This network service information is stored in the *interfaces* file.

The *interfaces* file

The *interfaces* file is usually named *interfaces*, *interface*, or *sql.ini*, depending on the operating system.

The *interfaces* file is like an address book. It lists the name and address of every known server. When you use a client program to connect to a server, the program looks up the server name in the *interfaces* file and then connects to the server using the address, as shown in Figure 1-1.

Figure 1-1: Connecting to Adaptive Server



The name, location, and contents of the *interfaces* file differ between operating systems. Also, the format of the Adaptive Server addresses in the *interfaces* file differs between network protocols.

When you install Adaptive Server, the installation program creates a simple *interfaces* file that you can use for local connections to Adaptive Server over one or more network protocols. As a System Administrator, it is your responsibility to modify the *interfaces* file and distribute it to users so that they can connect to Adaptive Server over the network. See the configuration documentation for your platform for information about the *interfaces* file for your platform.

Directory services

A directory service manages the creation, modification, and retrieval of network service information. Directory services are provided by platform or third-party vendors and must be purchased and installed separately from Adaptive Server. Two examples of directory services are NT Registry and Distributed Computing Environment (DCE).

The *$SYBASE/config/libtcl.cfg* file is a Sybase-supplied configuration file used by servers and clients to determine:

- Which directory service to use, and
- The location of the specified directory service driver.

If no directory services are installed or listed in the *libtcl.cfg* file, Adaptive Server defaults to the *interfaces* file for obtaining network service information.

The System Administrator must modify the *libtcl.cfg* file as appropriate for the operating environment.

Some directory services are specific to a given platform; others can be used on several different platforms. Because of the platform-specific nature of directory services, refer to the configuration documentation for your platform for detailed information on configuring for directory services.

LDAP as a directory service

Lightweight Directory Access Protocol (LDAP) is an industry standard for accessing directory services. Directory services allow components to look up information by a distinguished name (DN) from an LDAP server that stores and manages server, user, and software information that is used throughout the enterprise or over a network.

The LDAP server can be located on a different platform from the one on which Adaptive Server or the clients are running. LDAP defines the communication protocol and the contents of messages exchanged between clients and servers. Messages are operators, such as client requests for read, write and query, and server responses, including data-format information.

The LDAP server can store and retrieve information about:

- Adaptive Server, such as IP address, port number, and network protocol
- Security mechanisms and filters
- High availability companion server name
- Authentication information for user access to Adaptive Server

  You can authenticate users logging in to Adaptive Server through information stored in the *syslogins* directory or through a centralized LDAP server that enables a single login and password throughout the enterprise. See Chapter 10, “Managing Adaptive Server Logins, Database Users, and Client Connections,” for more information.

The LDAP server can be configured with these access restrictions:

- Anonymous authentication – all data is visible to any user.
- User name and password authentication – Adaptive Server uses the default user name and password from the file:

UNIX, u32-bit – *$SYBASE/$SYBASE_OCS/config/libtcl.cfg*

UNIX, 64-bit – *$SYBASE/$SYBASE_OCS/config/libtcl64.cfg*

NT – *%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg*

User name and password authentication properties establish and end a session connection to an LDAP server.

Note The default user name and password stored in *libtcl.cfg* and passed to the LDAP server for authentication purposes are distinct and different from those used to access Adaptive Server. The default user name and password allow access to the LDAP server for administrative tasks.

See "Creating and managing Adaptive Server logins using LDAP" on page 386 for information about using an LDAP server to store user accounts for accessing Adaptive Server.

When an LDAP server is specified in the *libtcl.cfg* or *libtcl64.cfg* file (collectively *libtcl*.cfg* file), the server information is accessible only from the LDAP server. Adaptive Server ignores the *interfaces* file.

If multiple directory services are supported in a server, then the order in which they are searched is specified in *libtcl*.cfg*. You cannot specify the search order with the dataserver command-line option.

Multiple directory services

Any type of LDAP service, whether it is an actual server or a gateway to other LDAP services, is called an LDAP server.

You can specify multiple directory services for high-availability failover protection in *libtcl*.cfg*. Not every directory service in the list needs to be an LDAP server.

In the following example, if the connection to *test:389* fails, the connection fails over to the DCE driver with the specified DIT base. If this also fails, a connection to the LDAP server on *huey:11389* is attempted. Different vendors employ different DIT base formats.

```
[DIRECTORY]
   ldap=libdldap.so ldap://test:389/dc=sybase,dc=com
   dce=libddce.so ditbase=/.:/subsys/sybase/dataservers
```

```
ldap=libdldap.so ldap://huey:11389/dc=sybase,dc=com
```

Note For more information, see the *Open Client Client-Library/C Programmer's Guide* and the *Open Client Client-Library/C Reference Manual*.

LDAP directory services versus the Sybase *interfaces* file

The LDAP driver implements directory services for use with an LDAP server. LDAP directories are an infrastructure that provide:

- A network-based alternative to the traditional Sybase *interfaces* file
- A single, hierarchical view of information, including users, software, resources, networks, files, and so on

Table 1-1 highlights the differences between the Sybase *interfaces* file and an LDAP server.

Table 1-1: interfaces file versus LDAP directory services

***interfaces* file**	**Directory services**
Platform-specific	Platform-independent
Specific to each Sybase installation	Centralized and hierarchical
Contains separate master and query entries	One entry for each server that is accessed by both clients and servers
Cannot store metadata about the server	Stores metadata about the server

Performance

Performance when using an LDAP server may be slower than when using an *interfaces* file because the LDAP server requires time to make a network connection and retrieve data. Since this connection is made when Adaptive Server is started, changes in performance will be seen at login time, if at all. During normal system load, the delay should not be noticeable. During high system load with many connections, especially repeated connections with short duration, the overall performance difference of using an LDAP server versus the traditional *interfaces* file might be noticeable.

Security features available in Adaptive Server

SQL Server version 11.0.6 passed the security evaluation by the National Security Agency (NSA) at the Class C2 criteria. (The requirements for the C2 criteria are given by the Department of Defense in DOD 52.00.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* [TCSEC], also known as the “Orange Book.”)

The configuration of SQL Server version 11.0.6 that was evaluated at the C2 security level by the NSA in 1996 on the HP 9000 HP-UX BLS, 9.09+ platform is referred to as the **evaluated configuration**. Certain features of SQL Server, such as remote procedures and direct updates to system tables, were excluded from the evaluated configuration. Notes in the Adaptive Server documentation indicate particular features that were not included in the evaluated configuration. For a complete list of features that were excluded from the evaluated configuration, see Appendix A in the *SQL Server Installation and Configuration Guide for HP 9000 HP-UX BLS, 9.09+*.

This version of Adaptive Server contains all of the security features included in SQL Server release 11.0.6 plus some new security features. Table 1-2 summarizes the major features.

Table 1-2: Major security features

Security feature	Description
Discretionary Access Controls (DAC)	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 upon an object owner’s discretion.
Identification and authentication controls	Ensures that only authorized users can log in to the system.
Division of roles	Allows you to grant privileged roles to specified users so that 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.”
Network-based security	Provides security services to authenticate users and protect data transmitted among machines on a network.
Auditing	Provides the capability to audit events such as logins, logouts, server boot operations, remote procedure calls, accesses to database objects, and all actions by a specific user or with a particular role active. In addition, Adaptive Server provides a single option to audit a set of server-wide security-relevant events.

CHAPTER 2 System and Optional Databases

This chapter describes the system databases that reside on all Adaptive Server systems. It also describes optional Sybase-supplied databases that you can install, and a database that Sybase Technical Support may install for diagnostic purposes.

Overview of system databases

When you install Adaptive Server, it includes these system databases:

- The master database
- The model database
- The system procedure database, sybsystemprocs
- The temporary database, tempdb

Optionally, you can install:

- The auditing database, sybsecurity
- The two-phase commit transaction database, sybsystemdb
- The sample databases, pubs2 and pubs3

- The dbcc database, dbccdb

For information about installing the master, model, sybsystemprocs, and tempdb databases, see the installation documentation for your platform. For information on installing dbccdb, see Chapter 26, “Checking Database Consistency.”

The master, model, and temporary databases reside on the device named during installation, which is known as the master device. The master database is contained entirely on the master device and cannot be expanded onto any other device. All other databases and user objects should be created on other devices.

Warning! Do not store user databases on the master device. Storing user databases on the master device makes it difficult to recover the system databases if they become damaged. Also, you will not be able to recover user databases stored on the master device.

You should install the sybsecurity and sybsystemdb databases on their own devices and segment. For more information, see the installation documentation for your platform.

You can install the sybsystemprocs database on a device of your choice. You may want to modify the installation scripts for pubs2 and pubs3 to share the device you create for sybsystemprocs.

The *installpubs2* and the *installpubs3* scripts do not specify a device in their create database statement, so they are created on the default device. At installation time, the master device is the default device. To change this, you can either edit the scripts or follow the instructions in Chapter 16, “Initializing Database Devices,” for information about adding more database devices and designating default devices.

master database

The master database controls the operation of Adaptive Server and stores information about all user databases and their associated database devices. Table 2-1 describes information the master database tracks.

Table 2-1: Information the master database tracks

Information	System table
User accounts	syslogins
Remote user accounts	sysremotelogins
Remote servers that this server can interact with	sysservers
Ongoing processes	sysprocesses
Configurable environment variables	sysconfigures
System error messages	sysmessages
Databases on Adaptive Server	sysdatabases
Storage space allocated to each database	sysusages
Tapes and disks mounted on the system	sysdevices
Active locks	syslocks
Character sets	syscharsets
Languages	syslanguages
Users who hold server-wide roles	sysloginroles
Server roles	syssrvroles
Adaptive Server engines that are online	sysengines

Because the master database stores information about user databases and devices, you must be in the master database in order to issue the create database, alter database, disk init, disk refit, disk reinit, and disk mirroring commands.

Controlling object creation in *master*

When you first install Adaptive Server, only a System Administrator can create objects in the master database, because the System Administrator implicitly becomes "dbo" of any database he or she uses. Any objects created on the master database should be used for the administration of the system as a whole. Permissions in master should remain set so that most users cannot create objects there.

Warning! Never place user objects in master. Storing user objects in master can cause the transaction log to fill quickly. If the transaction log runs out of space completely, you will not be able to use dump transaction commands to free space in master.

Another way to discourage users from creating objects in master is to change the default database for users (the database to which a user is connected when he or she logs in) with sp_modifylogin. See “Adding users to databases” on page 349 for more information.

If you create your own system procedures, create them in the sybsystemprocs database rather than in master.

Backing up *master* and keeping copies of system tables

To be prepared for hardware or software failure on Adaptive Server, the two most important housekeeping tasks are:

- Performing frequent backups of the master database and all user databases. See “Keep up-to-date backups of master” on page 38 for more information. See also Chapter 29, “Restoring the System Databases,” for an overview of the process for recovering the master database.
- Keeping a copy (preferably offline) of these system tables: sysusages, sysdatabases, sysdevices, sysloginroles, and syslogins. See “Keep offline copies of system tables” on page 39 for more information. If you have copies of these scripts, and a hard disk crash or other disaster makes your database unusable, you can use the recovery procedures described in Chapter 29, “Restoring the System Databases.” If you do not have current copies of your scripts, it will be much more difficult to recover Adaptive Server when the master database is damaged.

model database

Adaptive Server includes the model database, which provides a template, or prototype, for new user databases. Each time a user enters the create database command, Adaptive Server makes a copy of the model database and extends the new database to the size specified by the create database command.

Note A new database cannot be smaller than the model database.

The model database contains the required system tables for each user database. You can modify model to customize the structure of newly created databases—everything you do to model will be reflected in each new database. Some of the changes that System Administrators commonly make to model are:

- Adding user-defined datatypes, rules, or defaults.
- Adding users who should have access to all databases on Adaptive Server.
- Granting default privileges, particularly for "guest" accounts.
- Setting database options such as select into/bulkcopy/pllsort. The settings will be reflected in all new databases. Their original value in model is off. For more information about the database options, see Chapter 23, "Setting Database Options."

Typically, most users do not have permission to modify the model database. There is not much point in granting read permission either, since Adaptive Server copies its entire contents into each new user database.

The size of model cannot be larger than the size of tempdb. By default, the size of the model database is four allocation units. Adaptive Server displays an error message if you try to increase the size of model without making tempdb at least as large.

Note Keep a backup copy of the model database, and back up model with dump database each time you change it. In case of media failure, restore model as you would a user database.

sybsystemprocs database

Sybase system procedures are stored in the database sybsystemprocs. When a user in any database executes any stored procedure, Adaptive Server first looks for that procedure in the user's current database. If there is no procedure there with that name, Adaptive Server looks for it in sybsystemprocs. If there is no procedure in sybsystemprocs by that name, Adaptive Server looks for the procedure in master.

If the procedure modifies system tables (for example, sp_adduser modifies the sysusers table), the changes are made in the database from which the procedure was executed.

To change the default permissions on system procedures, you must modify those permissions in sybsystemprocs.

Note Any time you make changes to sybsystemprocs, you should back up the database.

tempdb database

Adaptive Server has a **temporary database**, tempdb. It provides a storage area for temporary tables and other temporary working storage needs. The space in tempdb is shared among all users of all databases on the server.

The default size of tempdb depends on the logical page size for your server, either 2, 4, 8, or 16K. Certain activities may make it necessary to increase the size of tempdb. The most common of these are:

- Large temporary tables.
- A lot of activity on temporary tables, which fills up the tempdb logs.
- Large sorts or many simultaneous sorts. Subqueries and aggregates with group by also cause some activity in tempdb.

You can increase the size of tempdb with alter database. tempdb is initially created on the master device. Space can be added from the master device or from any other database device.

Creating temporary tables

No special permissions are required to use tempdb, that is, to create temporary tables or to execute commands that may require storage space in the temporary database.

Create temporary tables either by preceding the table name in a create table statement with a pound sign (#) or by specifying the name prefix “tempdb..”.

Temporary tables created with a pound sign are accessible only by the current Adaptive Server session: users on other sessions cannot access them. These nonsharable, temporary tables are destroyed at the end of each session. The first 13 bytes of the table's name, including the pound sign (#), must be unique. Adaptive Server assigns the names of such tables a 17-byte number suffix. (You can see the suffix when you query tempdb..sysobjects.)

Temporary tables created with the "tempdb.." prefix are stored in tempdb and can be shared among Adaptive Server sessions. Adaptive Server does not change the names of temporary tables created this way. The table exists either until you restart Adaptive Server or until its owner drops it using drop table.

System procedures work on temporary tables, but only if you use them from tempdb.

If a stored procedure creates temporary tables, the tables are dropped when the procedure exits. Temporary tables can also be dropped explicitly before a session ends.

Warning! Do not create temporary tables with the "tempdb.." prefix from inside a stored procedure unless you intend to share those tables among other users and sessions.

Each time you restart Adaptive Server, it copies model to tempdb, which clears the database. Temporary tables are not recoverable.

sybsecurity database

The sybsecurity database contains the audit system for Adaptive Server. It consists of:

- The system tables, sysaudits_01, sysaudits_02, ... sysaudits_08, which contain the audit trail
- The sysauditoptions table, which contains rows describing the global audit options
- All other default system tables that are derived from model

The audit system is discussed in more detail in Chapter 11, "Auditing."

sybsystemdb database

The sybsystemdb database stores information about distributed transactions. Adaptive Server versions 12.0 and later can provide transaction coordination services for transactions that are propagated to remote servers using remote procedure calls (RPCs) or Component Integration System (CIS). Information about remote servers participating in distributed transactions is stored in the syscoordinations table.

Note Adaptive Server version 12.0 and later distributed transaction management services are available as a separately-licensed feature. You must purchase and install a valid license for Distributed Transaction Management before it can be used. See *Using Adaptive Server Distributed Transaction Management Features* and the installation guide for more information.

The sybsystemdb database also stores information about SYB2PC transactions that use the Sybase two-phase commit protocol. The spt_committab table, which stores information about and tracks the completion status of each two-phase commit transaction, is stored in the sybsystemdb database.

Two-phase commit transactions and how to create the sybsystemdb database is discussed in detail in the configuration documentation for your platform.

pubs2 and *pubs3* sample databases

Installing the pubs2 and pubs3 sample databases is optional. These databases are provided as a learning tool for Adaptive Server. The pubs2 sample database is used for most of the examples in the Adaptive Server documentation, except for examples, where noted, that use the pubs3 database. For information about installing pubs2 and pubs3, see the installation documentation for your platform. For information about the contents of these sample databases, see the *Transact-SQL User's Guide*.

Maintaining the sample databases

The sample databases contain a “guest” user that allows access to the database by any authorized Adaptive Server user. The “guest” user has been given a wide range of privileges in pubs2 and pubs3, including permissions to select, insert, update, and delete user tables. For more information about the “guest” user and a list of the guest permissions in pubs2 and pubs3, see Chapter 10, “Managing Adaptive Server Logins, Database Users, and Client Connections.”

The size of the pubs2 and pubs3 databases are determined by the size of the logical page size for your server, 2, 4, 8, and 16K. If possible, you should give each new user a clean copy of pubs2 and pubs3 so that she or he is not confused by other users’ changes. If you want to place pubs2 or pubs3 on a specific database device, edit the installation script before installing the database.

If space is a problem, you can instruct users to issue the begin transaction command before updating a sample database. After the user has finished updating one of the sample databases, he or she can issue the rollback transaction command to undo the changes.

pubs2 image data

Adaptive Server includes a script for installing image data in the pubs2 database (pubs3 does not use the image data). The image data consists of six pictures, two each in PICT, TIF, and Sun raster file formats. Sybase does not provide any tools for displaying image data. You must use the appropriate screen graphics tools to display the images after you extract them from the database.

See the the installation documentation for your platform for information about installing the image data in pubs2.

dbccdb database

dbcc checkstorage records configuration information for the **target database**, operation activity, and the results of the operation in the dbccdb database. Stored in the database are dbcc stored procedures for creating and maintaining dbccdb and for generating reports on the results of dbcc checkstorage operations. For more information, see Chapter 26, “Checking Database Consistency.”

sybdiag database

Sybase Technical Support may create the sybdiag database on your system for debugging purposes. This database holds diagnostic configuration data, and should not be used by customers.

CHAPTER 3

System Administration for Beginners

This chapter:

- Introduces new System Administrators to important topics
- Helps System Administrators find information in the Sybase documentation

Experienced administrators may also find this chapter useful for organizing their ongoing maintenance activities.

Logical page sizes

The logical page size is a server-wide setting. You cannot have databases with varyingly sized logical pages within the same server. Adaptive Server allows you to create master devices and databases with logical page sizes of 2K, 4K, 8K, or 16K, but a given server installation can have only one of these four logical page sizes. All databases in a server—and all objects in every database—use the same logical page size.

You select the page size when you create the master device with dataserver -z. All the logical pages of a server must be the same size. For instance, all the pages on a server with a logical page size of 4K must be 4K, even though you may not use some pages beyond the initial 2K.

For more information about the dataserver command see the *Utility Guide*. For more information about logical page sizes, see Chapter 18, “Configuring Memory.”

Using “test” servers

It is always best to install and use a “test” and/or “development” Adaptive Server, then remove it before you create the “production” server. Using a test server makes it easier to plan and test different configurations and less stressful to recover from mistakes. It is much easier to learn how to install and administer new features when there is no risk of having to restart a production server or re-create a production database.

If you decide to use a test server, we suggest that you do so from the point of installing or upgrading Adaptive Server through the process of configuring the server. It is in these steps that you make some of the most important decisions about your final production system. The following sections describe the ways in which using a test server can help System Administrators.

Understanding new procedures and features

Using a test server allows you to practice basic administration procedures before performing them in a production environment. If you are a new Adaptive Server administrator, many of the procedures discussed in this book may be unfamiliar to you, and it may take several attempts to complete a task successfully. However, even experienced administrators will benefit from practicing techniques that are introduced by new features in Adaptive Server.

Planning resources

Working with a test server helps you plan the final resource requirements for your system and helps you discover resource deficiencies that you might not have anticipated.

In particular, disk resources can have a dramatic effect on the final design of the production system. For example, you may decide that a particular database requires nonstop recovery in the event of a media failure. This would necessitate configuring one or more additional database devices to mirror the critical database. Discovering these resource requirements in a test server allows you to change the physical layout of databases and tables without affecting database users.

You can also use a test server to benchmark both Adaptive Server and your applications using different hardware configurations. This allows you to determine the optimal setup for physical resources at both the Adaptive Server level and the operating system level before bringing the entire system online for general use.

Achieving performance goals

Most performance objectives can be met only by carefully planning a database's design and configuration. For example, you may discover that the insert and I/O performance of a particular table is a bottleneck. In this case, the best course of action may be to re-create the table on a dedicated segment and partition the table. Changes of this nature are disruptive to a production system; even changing a configuration parameter may require you to restart Adaptive Server.

Installing Sybase products

The responsibility for installing Adaptive Server and other Sybase products is sometimes placed with the System Administrator. If installation is one of your responsibilities, use the following pointers to help you in the process.

Check product compatibility

Before installing new products or upgrading existing products, always read the release bulletin included with the products to understand any compatibility issues that might affect your system. Compatibility problems can occur between hardware and software and between different release levels of the same software. Reading the release bulletin in advance can save the time and guesswork of troubleshooting known compatibility problems.

Also, refer to the lists of known problems that are installed with Adaptive Server. See the release bulletin for more information.

Install or upgrade Adaptive Server

Read through the installation documentation for your platform before you begin a new installation or upgrade. You need to plan parts of the installation and configure the operating system *before* installing Adaptive Server. It is also helpful to consult with the operating system administrator to discuss operating system requirements for Adaptive Server. These requirements can include the configuration of memory, raw devices, asynchronous I/O, and other features, depending on the platform you use. Many of these tasks must be completed before you have begun the installation.

If you are upgrading a server, back up all data (including the master database, user databases, triggers, and system procedures) offline before you begin. After upgrading, immediately create a separate, full backup of your data, especially if there are incompatibilities between older dump files and the newer versions.

Install additional third-party software

Network protocols

Adaptive Server generally includes support for the network protocol(s) that are common to your hardware platform. If your network supports additional protocols, install the required protocol support.

Directory services

As an alternative to the Sybase *interfaces* file, you can use a directory service to obtain a server's address and other network information. Directory services are provided by platform or third-party vendors and must be purchased and installed separately from the installation of Adaptive Server. For more information on directory services currently supported by Adaptive Server, see the configuration documentation for your platform. See also "Directory services" on page 15.

Configure and test client connections

A successful client connection depends on the coordination of Adaptive Server, the client software, and network products. If you are using one of the network protocols installed with Adaptive Server, see the configuration documentation for your platform for information about testing network connections. If you are using a different network protocol, follow the instructions that are included with the network product. You can also use “ping” utilities that are included with Sybase connectivity products to test client connections with Adaptive Server. For a general description of how clients connect to Adaptive Server, see “Connecting to Adaptive Server” on page 14. See also the configuration documentation for your platform for details about the name and contents of the *interfaces* file.

Allocating physical resources

Allocating physical resources is the process of giving Adaptive Server the memory, disk space, worker processes, and CPU power required to achieve your performance and recovery goals. When installing a new server, every System Administrator must make decisions about resource utilization. You will also need to reallocate Adaptive Server’s resources if you upgrade your platform by adding new memory, disk controllers, or CPUs, or if the design of your database system changes. Or, early benchmarking of Adaptive Server and your applications can help you spot deficiencies in hardware resources that create performance bottlenecks.

See Chapter 15, “Overview of Disk Resource Issues,” in this manual to understand the kinds of disk resources required by Adaptive Server. See also Chapter 18, “Configuring Memory,” and Chapter 20, “Managing Multiprocessor Servers,” for information about memory and CPU resources.

The following sections provide helpful pointers in determining physical resource requirements.

Dedicated vs. shared servers

The first step in planning Adaptive Server resources is understanding the resources required by *other* applications running on the same machine. In most cases, System Administrators dedicate an entire machine for Adaptive Server use. This means that only the operating system and network software consume resources that otherwise might be reserved for Adaptive Server. On a shared system, other applications, such as Adaptive Server client programs or print servers, run on the same machine as Adaptive Server. It can be difficult to calculate the resources available to Adaptive Server on a shared system, because the types of programs and their pattern of use may change over time.

In either case, it is the System Administrator's responsibility to take into account the resources used by operating systems, client programs, windowing systems, and so forth when configuring resources for Adaptive Server. Configure Adaptive Server to use only the resources that are available to it. Otherwise, the server may perform poorly or fail to start.

Decision support and OLTP applications

Adaptive Server contains many features that optimize performance for OLTP, decision support, and mixed workload environments. However, you must determine in advance the requirements of your system's applications to make optimal use of these features.

For mixed workload systems, list the individual tables that you anticipate will be most heavily used for each type of application; this can help you achieve maximum performance for applications.

Advance resource planning

It is extremely important that you understand and plan resource usage in advance. In the case of disk resources, for example, after you initialize and allocate a device to Adaptive Server, that device cannot be used for any other purpose (even if Adaptive Server never fills the device with data). Likewise, Adaptive Server automatically reserves the memory for which it is configured, and this memory cannot be used by any other application.

The following suggestions can help you plan resource usage:

- For recovery purposes, it is *always* best to place a database's transaction log on a separate physical device from its data. See Chapter 21, "Creating and Managing User Databases."
- Consider mirroring devices that store mission-critical data. See Chapter 17, "Mirroring Database Devices." You may also consider using disk arrays and disk mirroring for Adaptive Server data if your operating system supports these features.
- If you are working with a test Adaptive Server, it is sometimes easier to initialize database devices as operating system files, rather than raw devices, for convenience. Adaptive Server supports either raw partitions or certified file systems for its devices.
- Keep in mind that changing configuration options can affect the way Adaptive Server consumes physical resources. This is especially true of memory resources. See Chapter 4, "Setting Configuration Parameters," for details about the amount of memory used by individual parameters.

Operating system configuration

Once you have determined the resources that are available to Adaptive Server and the resources you require, configure these physical resources at the operating system level:

- If you are using raw partitions, initialize the raw devices to the sizes required by Adaptive Server. Note that, if you initialize a raw device for Adaptive Server, that device cannot be used for any other purpose (for example, to store operating system files). Ask your operating system administrator for assistance in initializing and configuring raw devices to the required sizes.
- Configure the number of network connections. Make sure that the machine on which Adaptive Server runs can actually support the number of connections you configure. See your operating system documentation.
- Additional configuration may be required for your operating system and the applications that you use. Read the installation documentation for your platform to understand the Adaptive Server operating system requirements. Also read your client software documentation or consult with your engineers to understand the operating system requirements for your applications.

Backup and recovery

Making regular backups of your databases is crucial to the integrity of your database system. Although Adaptive Server automatically recovers from system crashes (for example, power outages) or server crashes, only *you* can recover from data loss caused by media failure. Follow the basic guidelines below for backing up your system.

The following chapters describe how to develop and implement a backup and recovery plan:

- Chapter 27, “Developing a Backup and Recovery Plan”
- Chapter 28, “Backing Up and Restoring User Databases”
- Chapter 29, “Restoring the System Databases”
- Chapter 31, “Managing Free Space with Thresholds”

Keep up-to-date backups of master

Backing up the master database is the cornerstone of any backup and recovery plan. The master database contains details about the structure of your entire database system. Its keeps track of the Adaptive Server databases, devices, and device fragments that make up those databases. Because Adaptive Server needs this information during recovery, it is crucial to maintain an up-to-date backup copy of the master database at all times.

To ensure that your backup of master is always up to date, back up the database after each command that affects disks, storage, databases, or segments. This means you should back up master after performing any of the following procedures:

- Creating or deleting databases
- Initializing new database devices
- Adding new dump devices
- Using any device mirroring command
- Creating or dropping system stored procedures, if they are stored in master
- Creating, dropping, or modifying a segment
- Adding new Adaptive Server logins

To back up master to a tape device, start isql and enter the command, where *tape_device* is the name of the tape device (for example, */dev/rmt0*):

```
dump database master to "tape_device"
```

Keep offline copies of system tables

In addition to backing up master regularly, keep offline copies of the contents of the following system tables: sysdatabases, sysdevices, sysusages, sysloginroles, and syslogins. Do this by using the bcp utility described in the *Utility Guide*, and by storing a printed copy of the contents of each system table. You can create a printed copy by printing the output of the following queries:

```
select * from sysusages order by vstart
select * from sysdatabases
select * from sysdevices
select * from sysloginroles
select * from syslogins
```

If you have copies of these tables, and a hard disk crash or some other disaster makes your database unusable, you will be able to use the recovery procedures described in Chapter 29, "Restoring the System Databases."

You should also keep copies of all data definition language (DDL) scripts for user objects, as described under "Keeping records" on page 42.

Automate backup procedures

Creating an automated backup procedure takes the guesswork out of performing backups and makes the procedure easier and quicker to perform. Automating backups can be as simple as using an operating system script or a utility (for example, the UNIX cron utility) to perform the necessary backup commands. Or you can automate the procedure further using thresholds, which are discussed in Chapter 31, "Managing Free Space with Thresholds."

v **Creating an automated backup procedure**

Although the commands required to create an automated script vary, depending on the operating system you use, all scripts should accomplish the same basic steps:

1 Start isql and dump the transaction log to a holding area (for example, a temporary file).

2. Rename the dump file to a name that contains the dump date, time, and database name.
3. Make a note about the new backup in a history file.
4. Record any errors that occurred during the dump in a separate error file.
5. Automatically send mail to the System Administrator for any error conditions.

Although the commands required to create an automated script vary, depending on the operating system you use, all scripts should accomplish the same basic steps:

Verify data consistency before backing up a database

Having backups of a database sometimes is not enough—you must have consistent, *accurate* backups (especially for master). If you back up a database that contains internal errors, the database will have the same errors when you restore it.

Using the dbcc commands, you can check a database for errors before backing it up. Always use dbcc commands to verify the integrity of a database before dumping it. If dbcc detects errors, correct them before dumping the database.

Over time, you can begin to think of running dbcc as insurance for your databases. If you discovered few or no errors while running dbcc in the past, you may decide that the risk of database corruption is small and that dbcc needs to be run only occasionally. Or, if the consequences of losing data are too high, you should continue to run dbcc commands each time you back up a database.

Note For performance considerations, many sites choose to run dbcc checks outside of peak hours or on separate servers.

See Chapter 26, “Checking Database Consistency,” for information about the dbcc command.

Monitor the log size

When the transaction log becomes nearly full, it may be impossible to use standard procedures to dump transactions and reclaim space. The System Administrator should monitor the log size and perform regular transaction log dumps (in addition to regular database dumps) to make sure this situation never occurs. Use the preferred method of setting up a threshold stored procedure that notifies you (or dumps the log) when the log reaches a certain capacity. See Chapter 31, "Managing Free Space with Thresholds," for information about using threshold procedures. It is also good to dump the transaction log just prior to doing a full database dump in order to shorten the time required to dump and load the database.

You can also monitor the space used in the log segment manually by using the sp_helpsegment stored procedure, as described under "Getting information about segments" on page 735.

Ongoing maintenance and troubleshooting

In addition to making regularly scheduled backups, the System Administrator performs the following maintenance activities throughout the life of a server.

Starting and stopping Adaptive Server

Most System Administrators automate the procedure for starting Adaptive Server to coincide with the start-up of the server machine. This can be accomplished by editing operating system start-up scripts or through other operating system procedures. See the configuration documentation for your platform to determine how to start and stop Adaptive Server.

Viewing and pruning the error log

You should examine the contents of the error log on a regular basis to determine if any serious errors have occurred. You can also use operating system scripts to scan the error log for particular messages and to notify the System Administrator when specific errors occur. Checking the error log regularly helps you determine whether there are continuing problems of the same nature or whether a particular database device is going bad. See Chapter 5, “Diagnosing System Problems,” for more information about error messages and their severity.

The error log file can grow large over time, since Adaptive Server appends informational and status messages to it each time it starts up. You can periodically “prune” the log file by opening the file and deleting old records. Keeping the log file to a manageable size saves disk space and makes it easier to locate current errors.

Keeping records

Keeping records about your Adaptive Server system is an important part of your job as a System Administrator. Accurate records of changes and problems that you have encountered can be a valuable reference when you are contacting Sybase Technical Support or recovering databases. More important, they can provide vital information for administrators who manage the Adaptive Server system in your absence. The following sections describe the kinds of records that are most valuable to maintain.

Contact information

Maintain a list of contact information for yourself as well as the System Security Officer, operator, and database owners on your system. Also, record secondary contacts for each role. Make this information available to all Adaptive Server users so that the appropriate contacts receive enhancement requests and problem reports.

Configuration information

Ideally, you should create databases, create database objects, and configure Adaptive Server using script files that you later store in a safe place. Storing the script files use makes it possible to re-create your entire system in the event of a disaster. It also allows you to re-create database systems quickly on new hardware platforms for evaluation purposes. If you use a third-party tool to perform system administration, remember to generate equivalent scripts after performing administration tasks.

Consider recording the following kinds of information:

- Commands used to create databases and database objects (DDL scripts)
- Commands that add new Adaptive Server logins and database users
- The current Adaptive Server configuration file, as described in "Using sp_configure with a configuration file" on page 57
- The names, locations, and sizes of all files and raw devices initialized as database devices

It is also helpful to maintain a dated log of all changes to the Adaptive Server configuration. Mark each change with a brief description of when and why you made the change, as well a summary of the end result.

Maintenance schedules

Keep a calendar of regularly scheduled maintenance activities. Such a calendar should list any of the procedures you perform at your site:

- Using dbcc to check database consistency
- Backing up user and system databases
- Monitoring the space left in transaction logs (if this is not done automatically)
- Dumping the transaction log
- Examining the error log contents for Adaptive Server, Backup Server™, and Adaptive Server Monitor™
- Running the update statistics command (see Chapter 4, "Using the set statistics Commands," in *Performance and Tuning: Monitoring and Analyzing*)
- Examining auditing information, if the auditing option is installed

- Recompiling stored procedures
- Monitoring the resource utilization of the server machine

System information

Record information about the hardware and operating system on which you run Adaptive Server. This can include:

- Copies of operating system configuration files or start-up files
- Copies of network configuration files (for example, the *hosts* and *services* files)
- Names and permissions for the Adaptive Server executable files and database devices
- Names and locations of the tape devices used for backups
- Copies of operating system scripts or programs for automated backups, starting Adaptive Server, or performing other administration activities

Disaster recovery plan

Consolidate the basic backup and recovery procedures, the hints provided in "Backup and recovery" on page 38, and your personal experiences in recovering data into a concise list of recovery steps tailored to your system. This can be useful to both yourself and to other System Administrators who may need to recover a production system in the event of an emergency.

Getting more help

The amount of new information that System Administrators must learn may seem overwhelming. There are several software tools that can help you learn and facilitate basic administration tasks. These include Adaptive Server Monitor, used for monitoring server performance and other activities, and Sybase Central, which simplifies many administration tasks. Also available are many third-party software packages designed to help System Administrators manage daily maintenance activities.

CHAPTER 4

Setting Configuration Parameters

This chapter describes the Adaptive Server configuration parameters. A configuration parameter is a user-definable setting that you set with the system procedure sp_configure. Configuration parameters are used for a wide range of services, from basic to specific server operations, and for performance tuning.

Adaptive Server configuration parameters

The following lists the Adaptive Server configuration parameters alphabetically.

- “abstract plan cache” on page 148
- “abstract plan dump” on page 148
- “abstract plan load” on page 148
- “abstract plan replace” on page 149
- “additional network memory” on page 142
- “allocate max shared memory” on page 139
- “allow backward scans” on page 149
- “allow nested triggers” on page 150
- “allow procedure grouping” on page 183
- “allow remote access” on page 124
- “allow resource limits” on page 150
- “allow sendmsg” on page 125
- “allow sql server async i/o” on page 85
- “allow resource limits” on page 150

- “allow updates to system tables” on page 151
- “auditing” on page 183
- “audit queue size” on page 187
- “check password for digit” on page 185
- “cis bulk insert array size” on page 81
- “cis bulk insert batch size” on page 81
- “cis cursor rows” on page 82
- “cis packet size” on page 82
- “cis rpc handling” on page 83
- “configuration file” on page 102
- “cpu accounting flush interval” on page 152
- “cpu grace time” on page 153
- “current audit table” on page 188
- “deadlock checking period” on page 110
- “deadlock retries” on page 111
- “default character set id” on page 106
- “default database size” on page 154
- “default exp_row_size percent” on page 155
- “default fill factor percent” on page 154
- “default language id” on page 106
- “default network packet size” on page 125
- “default sortorder id” on page 107
- “default unicode sortorder” on page 194
- “disable character set conversions” on page 107
- “disk i/o structures” on page 86
- “dtm detach timeout period” on page 88
- “dtm lock timeout period” on page 89
- “dynamic allocation on demand” on page 140

- “enable full-text search” on page 84
- “enable cis” on page 83
- “enable DTM” on page 90
- “enable housekeeper GC” on page 161
- “enable HA” on page 160
- “enable java” on page 103
- “enable enterprise java beans” on page 104
- “enable file access” on page 83
- “enable full-text search” on page 84
- “enable ldap user auth” on page 194
- “enable rep agent threads” on page 147
- “enable row level access control” on page 157
- “enable ssl” on page 189
- “enable sort-merge joins and JTC” on page 156
- “enable surrogate processing” on page 195
- “enable unicode conversion” on page 195
- “enable unicode normalization” on page 196
- “enable xact coordination” on page 91
- “esp execution priority” on page 99
- “esp execution stacksize” on page 99
- “esp unload dll” on page 100
- “event buffers per engine” on page 157
- “event log computer name (Windows NT only)” on page 97
- “event logging (Windows NT only)” on page 97
- “executable codesize + overhead” on page 116
- “global async prefetch limit” on page 75
- “global cache partition number” on page 76
- “heap memory per user” on page 143

- "housekeeper free write percent" on page 158
- "i/o accounting flush interval" on page 163
- "i/o polling process count" on page 164
- "identity burning set factor" on page 161
- "identity grab size" on page 162
- "license information" on page 182
- "lock address spinlock ratio" on page 108
- "lock hashtable size" on page 113
- "lock shared memory" on page 143
- "lock scheme" on page 114
- "lock spinlock ratio" on page 112
- "lock table spinlock ratio" on page 115
- "lock wait period" on page 114
- "log audit logon failure" on page 98
- "log audit logon success" on page 98
- "max async i/os per engine" on page 132
- "max async i/os per server" on page 133
- "max cis remote connections" on page 84
- "max memory" on page 141
- "max network packet size" on page 126
- "max number network listeners" on page 129
- "max online engines" on page 146
- "max parallel degree" on page 137
- "max scan parallel degree" on page 138
- "max SQL text monitored" on page 144
- "maximum dump conditions" on page 167
- "maximum failed logins" on page 183
- "minimum password length" on page 185

- “memory alignment boundary” on page 76
- “memory per worker process” on page 139
- “msg confidentiality reqd” on page 189
- “msg integrity reqd” on page 190
- “number of alarms” on page 168
- “number of aux scan descriptors” on page 168
- “number of devices” on page 86
- “number of dtx participants” on page 92
- “number of engines at startup” on page 145
- “number of index trips” on page 77
- “number of large i/o buffers” on page 71
- “number of locks” on page 109
- “number of mailboxes” on page 171
- “number of messages” on page 172
- “number of oam trips” on page 78
- “number of open databases” on page 117
- “number of open indexes” on page 119
- “number of open objects” on page 120
- “number of pre-allocated extents” on page 172
- “number of remote connections” on page 129
- “number of remote logins” on page 129
- “number of remote sites” on page 130
- “number of sort buffers” on page 173
- “number of user connections” on page 197
- “number of worker processes” on page 136
- “open index hash spinlock ratio” on page 122
- “open index spinlock ratio” on page 123
- “open object spinlock ratio” on page 123

- “o/s file descriptors” on page 134
- “page lock promotion HWM” on page 165
- “page lock promotion LWM” on page 166
- “page lock promotion PCT” on page 167
- “page utilization percent” on page 87
- “partition groups” on page 173
- “partition spinlock ratio” on page 174
- “permission cache entries” on page 199
- “print deadlock information” on page 174
- “print recovery information” on page 71
- “procedure cache size” on page 79
- “read committed with lock” on page 115
- “recovery interval in minutes” on page 72
- “remote server pre-read packets” on page 130
- “row lock promotion HWM” on page 180
- “row lock promotion LWM” on page 180
- “row lock promotion PCT” on page 181
- “runnable process search count” on page 175
- “sampling percent” on page 102
- “secure default login” on page 190
- “select on syscomments.text column” on page 191
- “shared memory starting address” on page 135
- “size of auto identity column” on page 176
- “size of global fixed heap” on page 105
- “size of process object heap” on page 105
- “size of shared class heap” on page 105
- “size of unilib cache” on page 196
- “SQL Perfmon Integration (Windows NT only)” on page 177

- “sql server clock tick length” on page 177
- “stack guard size” on page 200
- “stack size” on page 202
- “start mail session (Windows NT only)” on page 100
- “strict dtm enforcement” on page 93
- “suspend audit when device full” on page 191
- “syb_sendmsg port number” on page 131
- “systemwide password expiration” on page 192
- “lock table spinlock ratio” on page 115
- “tape retention in days” on page 75
- “tcp no delay” on page 131
- “text prefetch size” on page 178
- “time slice” on page 179
- “total data cache size” on page 80
- “total logical memory” on page 145
- “total physical memory” on page 144
- “txn to pss ratio” on page 94
- “unified login required (Windows NT only)” on page 193
- “upgrade version” on page 179
- “user log cache size” on page 204
- “user log cache spinlock ratio” on page 205
- “use security services (Windows NT only)” on page 193
- “xact coordination interval” on page 95
- “xp_cmdshell context” on page 101

What are configuration parameters?

Configuration parameters are user-definable settings that control various aspects of Adaptive Server's behavior. Adaptive Server supplies default values for all configuration parameters. You can use configuration parameters to tailor Adaptive Server for an installation's particular needs.

Read this chapter carefully to determine which configuration parameters you should reset to optimize server performance. Also, see the *Performance and Tuning Guide* for further information on using sp_configure to tune Adaptive Server.

Warning! Change configuration parameters with caution. Arbitrary changes in parameter values can adversely affect Adaptive Server performance and other aspects of server operation.

The Adaptive Server configuration file

Adaptive Server stores the values of configuration parameters in a configuration file, which is an ASCII text file. When you install a new Adaptive Server, your parameters are set to the default configuration; the default name of the file is *server_name.cfg*, and the default location of the file is the Sybase installation directory (*$SYBASE*). When you change a configuration parameter, Adaptive Server saves a copy of the old configuration file as *server_name.001*, *server_name.002*, and so on. Adaptive Server writes the new values to the file *server_name.cfg* or to a file name you specify at start-up.

How to modify configuration parameters

You set or change configuration parameters in one of the following ways:

- By executing the system procedure sp_configure with the appropriate parameters and values,
- By editing your configuration file and then invoking sp_configure with the configuration file option, or
- By specifying the name of a configuration file at start-up.

Configuration parameters are either *dynamic* or *static*. Dynamic parameters go into effect as soon as you execute sp_configure. Static parameters require Adaptive Server to reallocate memory, so they take effect only after Adaptive Server has been restarted. The description of each parameter indicates whether it is static or dynamic. Adaptive Server writes the new value to the system table sysconfigures and to the configuration file when you change the value, not when you restart Adaptive Server. The current configuration file and sysconfigures reflect configured values, not run values. The system table syscurconfigs reflects current run values of configuration parameters.

Who can modify configuration parameters?

The roles required for using sp_configure are as follows:

- Any user can execute sp_configure to display information about parameters and their current values.
- Only a System Administrator and System Security Officer can execute sp_configure to modify configuration parameters.
- Only a System Security Officer can execute sp_configure to modify values for:

  allow procedure grouping
  allow remote access
  allow procedure grouping
  allow updates to system tables
  audit queue size
  auditing
  current audit table
  enable ssl
  msg confidentiality reqd
  msg integrity reqd
  secure default login
  select on syscomments.text column
  suspend audit when device full
  systemwide password expiration
  secure default login
  unified login required (Windows NT only)
  use security services (Windows NT only)

Unit specification using *sp_configure*

sp_configure allows you to specify the value for configuration parameters in unit specifiers. The unit specifiers are p or P for pages, m or M for megabytes, and g or G for gigabytes. If you do not specify a unit, and you are configuring a parameter that controls memory, Adaptive Server uses the logical page size for the basic unit.

The syntax to indicate a particular unit specification is:

```
sp_configure "parameter name", 0, "p|P|k|K|m|M|g|G"
```

You must include the “0” as a placeholder.

You can use this unit specification to configure any parameter. For example, when setting number of locks to 1024 you can enter:

```
sp_configure "number of locks", 1024
```

or:

```
sp_configure "number of locks", 0, 1K
```

This functionality will not change the way in which Adaptive Server reports sp_configure output.

Note When you are configuring memory related parameters, only use the P (pagesize) parameter for your unit specification. If you use any other parameter to configure memory related parameters, Adaptive Server may issue an arithmetic overflow error message.

Getting help information on configuration parameters

Use either sp_helpconfig or sp_configure to get information on a particular configuration parameter. For example:

```
sp_helpconfig "number of open"
```

```
Configuration option is not unique.
option_name                    config_value  run_value
------------------------------ ------------ -----------
number of open databases                 12          12
number of open indexes                  500         500
number of open objects                  500         500
```

```
sp_helpconfig "number of open indexes"
```

```
number of open indexes sets the maximum number of indexes that can be open at
one time on SQL Server. The default value is 500.
Minimum Value Maximum Value Default Value Current Value Memory Used
------------- ------------- ------------- ------------- -----------
          100    2147483647           500           500         208


                       sp_configure "number of open indexes"

Parameter Name           Default  Memory Used  Config Value  Run Value
-----------------------  -------  -----------  ------------  ---------
number of open indexes       500          208           500        500
```

For more information, see "Using sp_helpconfig to get help on configuration parameters" on page 603.

Using *sp_configure*

sp_configure displays and resets configuration parameters. You can restrict the number of parameters displayed by sp_configure using sp_displaylevel to set your display level to one of three values:

- Basic
- Intermediate
- Comprehensive

For information about display levels, see "User-defined subsets of the parameter hierarchy: display levels" on page 63. For information about sp_displaylevel, see the *Reference Manual: Stored Procedures*.

Table 4-1 describes the syntax for sp_configure. The information in the "Effect" column assumes that your display level is set to "comprehensive."

Table 4-1: sp_configure syntax

Command	Effect
sp_configure	Displays all configuration parameters by group, their current values, their default values, the value to which they have most recently been set, and the amount of memory used by this particular setting.
sp_configure "*parameter*"	Displays current value, default value, most recently changed value, and amount of memory used by setting for all parameters matching parameter.
sp_configure "*parameter*", *value*	Resets *parameter* to *value*.

Command	Effect
sp_configure "*parameter*", 0, "default"	Resets parameter to its default value.
sp_configure "*group_name*"	Displays all configuration parameters in *group_name*, their current values, their default values, the values to which they were recently set, and the amount of memory used by each setting.
sp_configure "configuration file", 0, "*sub_command*", "*file_name*"	Sets configuration parameters from the configuration file. See "Using sp_configure with a configuration file" on page 57 for descriptions of the parameters.

Syntax elements

The commands in Table 4-1 use the following variables:

- *parameter* – is any valid Adaptive Server configuration parameter or parameter substring.
- *value* – is any integer within the valid range for that parameter. (See the descriptions of the individual parameters for valid range information.) Parameters that are toggles have only two valid values: 1 (on) and 0 (off).
- *group_name* – is the name of any group in the parameter hierarchy.

Parameter parsing

sp_configure parses each parameter (and parameter name fragment) as "*%parameter%*". A string that does not uniquely identify a particular parameter returns values for all parameters matching the string.

The following example returns values for all configuration parameters that include "lock," such as lock shared memory, number of locks, lock promotion HWM, server clock tick length, print deadlock information, and deadlock retries:

```
sp_configure "lock"
```

Note If you attempt to set a parameter value with a nonunique parameter name fragment, sp_configure returns the current values for all parameters matching the fragment and asks for a unique parameter name.

Using *sp_configure* with a configuration file

You can configure Adaptive Server either interactively, by using sp_configure as described above, or noninteractively, by instructing Adaptive Server to read values from an edited or restored version of the configuration file.

The benefits of using configuration files include:

- You can replicate a specific configuration across multiple servers by using the same configuration file.
- You can use a configuration file as a baseline for testing configuration values on your server.
- You can use a configuration file to do validation checking on parameter values before actually setting the values.
- You can create multiple configuration files and switch between them as your resource needs change.

You can make a copy of the configuration file using sp_configure with the parameter "configuration file" and then edit the file at the operating system level. Then, you can use sp_configure with the parameter "configuration file" to instruct Adaptive Server to read values from the edited file. Or you can specify the name of the configuration file at start-up.

For information on editing the file, see "Editing the configuration file" on page 59. For information on specifying the name of the configuration file at start-up, see "Starting Adaptive Server with a configuration file" on page 61.

Naming tips for the configuration file

Each time you modify a configuration parameter with sp_configure, Adaptive Server creates a copy of the outdated configuration file, using the naming convention *server_name.001*, *server_name.002*, *server_name.003*...*server_name.999*.

If you want to work with a configuration file with a name other than the default name, and you keep the *server_name* part of the file name, be sure to include at least one alphabetic character in the extension. Alternatively, you can change the *server_name* to part of the file name. Doing this avoids confusion with the backup configuration files generated by Adaptive Server when you modify a parameter.

Using *sp_configure* to read or write the configuration file

The syntax for using the configuration file option with sp_configure is:

```
sp_configure "configuration file", 0, "subcommand", "file_name"
```

where:

- "configuration file" – including quotes, specifies the configuration file parameter.
- 0 – must be included as the second parameter to sp_configure for backward compatibility.
- "*subcommand*" – is one of the commands described below.
- *file_name* – specifies the configuration file you want to use in conjunction with any *subcommand*. If you do not specify a directory as part of the file name, the directory where Adaptive Server was started is used.

Parameters for using configuration files

The four parameters described below can be used with configuration files.

write – creates *file_name* from the current configuration. If *file_name* already exists, a message is written to the error log; the existing file is renamed using the convention *file_name.001*, *file_name.002,* and so on. If you have changed a static parameter, but you have not restarted your server, write gives you the *currently running value* for that parameter. If you do not specify a directory with *file_name,* the file is written to the directory from which Adaptive Server was started.

read – performs validation checking on values contained in *file_name* and reads those values that pass validation into the server. If any parameters are missing from *file_name,* the current values for those parameters are used.

If the value of a static parameter in *file_name* is different from its current running value, read fails and a message is printed. However, validation is still performed on the values in *file_name*.

verify – performs validation checking on the values in *file_name*. This is useful if you have edited the configuration file, as it prevents you from attempting to configure your server with invalid configuration values.

restore – creates *file_name* with the most recently configured values. If you have configured static parameters to new values, this subcommand will write the configured, not the currently running, values to the file. This is useful if all copies of the configuration file have been lost and you need to generate a new copy. If you do not specify a directory with *file_name*, the file is written to the directory from which Adaptive Server was started.

Examples

Example 1 Performs validation checking on the values in the file *srv.config* and reads the parameters that pass validation into the server. Current run values are substituted for values that do not pass validation checking:

```
sp_configure "configuration file", 0, "read", "srv.config"
```

Example 2 Creates the file *my_server.config* and writes the current configuration values the server is using to that file:

```
sp_configure "configuration file", 0, "write", "my_server.config"
```

Example 3 Runs validation checking on the values in the file *generic.config*:

```
sp_configure "configuration file", 0, "verify", "generic.config"
```

Example 4 Writes configured values to the file *restore.config*:

```
sp_configure "configuration file", 0, "restore", "restore.config"
```

Editing the configuration file

The configuration file is an operating system ASCII file that you can edit with any text editor that can save files in ASCII format. The syntax for each parameter is:

```
parameter_name={value | DEFAULT}
```

where:

- *parameter_name* – is the name of the parameter you want to specify
- *value* – is the numeric value for set *parameter_name*
- “DEFAULT” – specifies that you want to use the default value for *parameter_name*

Examples

Example 1 The following example specifies that the transaction can retry to acquire a lock one time when deadlocking occurs during an index page split or shrink:

```
cpu accounting flush interval=DEFAULT
```

Example 2 The following example specifies that the default value for the parameter cpu accounting flush interval should be used:

```
deadlock retries = 1
```

When you edit a configuration file, your edits are not validated until you check the file using the verify option, read the file with the read option, or restart Adaptive Server with that configuration file.

If all your configuration files are lost or corrupted, you can re-create one from a running server by using the restore subcommand and specifying a name for the new file. The parameters in the new file will be set to the values with which your server is currently running.

Permissions for configuration files

Configuration files are nonencrypted ASCII text files. By default, they are created with read and write permissions set for the file owner and read permission set for all other users. If you created the configuration file at the operating system level, you are the file owner; if you created the configuration file from Adaptive Server, using the write or restore parameter, the file owner is the user who started Adaptive Server. Usually, this is the user "sybase." To restrict access to configuration files, use your operating system's file permission command to set read, write, and execute permissions as appropriate.

Note You need to set permissions accordingly on *each* configuration file created.

Backing up configuration files

Configuration files are not automatically backed up when you back up the master database. They are operating system files, and you should back them up in the same way you back up your other operating system files.

Checking the name of the configuration file currently in use

The output from sp_configure truncates the name of the configuration file due to space limitations. To see the full name of the configuration file, use:

```
select s1.value2
from syscurconfigs s1, sysconfigures s2
where s1.config = s2.config
and s2.name = "configuration file"
```

Starting Adaptive Server with a configuration file

By default, Adaptive Server reads the configuration file *server_name.cfg* in the start-up directory when it starts. If this file does not exist, it creates a new file and uses Adaptive Server defaults for all values.

You can start Adaptive Server with a specified configuration file. For more information, see the *Utility Guide*.

If the configuration file you specify does not exist, Adaptive Server prints an error message and does not start.

If the command is successful, the file *server_name.bak* is created. This file contains the configuration values stored in sysconfigures prior to the time sysconfigures was updated with the values read in from the configuration file you specified. This file is overwritten with each subsequent start-up.

Configuration file errors

When there are errors in the configuration file, Adaptive Server may not start or may use default values.

Adaptive Server uses default values if:

- There are illegal values. For example, if a parameter requires a numeric value, and the configuration file contains a character string, Adaptive Server uses the default value.
- Values are below the minimum allowable value.

The parameter hierarchy

Configuration parameters are grouped according to the area of Adaptive Server behavior they affect. This makes it easier to identify all parameters that you might need to tune improve a particular area of Adaptive Server performance.

The groups are:

- Backup and recovery
- Cache manager
- Component Integration Services administration
- Disk I/O
- DTM administration

- Error log
- Extended stored procedures
- General information
- Java services
- Languages
- Lock Manager
- Memory use
- Metadata caches
- Network communication
- O/S resources
- Parallel queries
- Physical memory
- Processors
- RepAgent thread administration
- SQL server administration
- Security related
- User environment

Although each parameter has a primary group to which it belongs, many have secondary groups to which they also belong. For example, number of remote connections belongs primarily to the Network Communication group, but it also belongs secondarily to the Adaptive Server Administration group and the Memory Use group. This reflects the fact that some parameters have implications for a number of areas of Adaptive Server behavior. sp_configure displays parameters in all groups to which they belong.

The syntax for displaying all groups and their associated parameters, and the current values for the parameters, is:

```
sp_configure
```

Note The number of parameters sp_configure returns depends on the value to which you have your display level set. See "User-defined subsets of the parameter hierarchy: display levels" on page 63 for further information about display levels.

The following is the syntax for displaying a particular group and its associated parameter, where *group_name* is the name of the group you are interested in:

```
sp_configure "group_name"
```

For example, to display the Disk I/O group, type:

```
sp_configure "Disk I/O"
Group: Disk I/O

Parameter Name           Default Memory Used Config Value Run Value
--------------           ------- ----------- ------------ ---------
allow sql server async i/o     1           0            1         1
disk i/o structures          256           0          256       256
number of devices             10           0           10        10
page utilization percent      95           0           95        95
```

Note If the server uses a case-insensitive sort order, sp_configure with no parameters returns a list of all configuration parameters and groups in alphabetical order with no grouping displayed.

User-defined subsets of the parameter hierarchy: display levels

Depending on your use of Adaptive Server, you may need to adjust some parameters more frequently than others. You may find it is easier to work with a subset of parameters than having to see the entire group when you are working with only a few. You can set your display level to one of three values to give you the subset of parameters that best suits your working style.

The default display level is “comprehensive.” When you set your display level, the setting persists across multiple sessions. However, you can reset it at any time to see more or fewer configuration parameters.

- “Basic” shows just the most basic parameters. It is appropriate for very general server tuning.
- “Intermediate” shows you parameters that are somewhat more complex, in addition to the “basic” parameters. This level is appropriate for a moderately complex level of server tuning.
- “Comprehensive” shows you all the parameters, including the most complex ones. This level is appropriate for users doing highly detailed server tuning.

The syntax for showing your current display level is:

```
sp_displaylevel
```

The following is the syntax for setting your display level, where *user_name* is your Adaptive Server login name:

```
sp_displaylevel user_name [, basic | intermediate | comprehensive]
```

The effect of the display level on *sp_configure* output

If your display level is set to either "basic" or "intermediate," sp_configure returns only a subset of the parameters that are returned when your display level is set to "comprehensive." For instance, if your display level is set to "intermediate," and you want to see the parameters in the Languages group, type:

```
sp_configure "Languages"
```

The output would look like this:

```
sp_configure
Group: Languages

Parameter Name         Default Memory Used Config Value Run Value Unit Type
---------------        ------- ----------- ------------ --------- ---- ----
default character set    1          0            1          1     id   static
default language id      0          0            0         0      id   dyna
. . .
```

However, this is only a subset of the parameters in the Languages group, because some parameters in that group are displayed only at the "comprehensive" level.

The *reconfigure* command

Pre-11.0 SQL Server versions required you to execute reconfigure after executing sp_configure. Beginning with SQL Server version 11.0, this was no longer required. The reconfigure command still exists, but it does not have any effect. It is included in this version of Adaptive Server so you can run pre-11.0 SQL scripts without modification.

Scripts using reconfigure will still run in the current version, but you should change them at your earliest convenience because reconfigure will not be supported in future versions of Adaptive Server.

Performance tuning with *sp_configure* and *sp_sysmon*

sp_sysmon monitors Adaptive Server performance and generates statistical information that describes the behavior of your Adaptive Server system. See the *Performance and Tuning Guide* for more information.

You can run sp_sysmon before and after using sp_configure to adjust configuration parameters. The output gives you a basis for performance tuning and lets you observe the results of configuration changes.

This chapter includes cross-references to the *Performance and Tuning Guide* for the sp_configure parameters that can affect Adaptive Server performance.

Output from *sp_configure*

The sample output below shows the kind of information sp_configure prints if you have your display level set to "comprehensive" and you execute it with no parameters. The values it prints will vary, depending on your platform and on what values you have already changed.

```
sp_configure
Group: Configuration Options

Group: Backup/Recovery

Parameter Name        Default Memory Used Config Value Run Value Unit Type
--------------        ------- ----------- ------------ --------- ---- ----
allow remote access         1           0            1         1    switch dyn
print recovery info         0           0            0         0    switch dyn
recovery interval in m      5           0            5         5    minutes dyn
...
```

Note All configuration groups and parameters will appear in output if your display level is set to "comprehensive."

Where:

- The "Default" column displays the value Adaptive Server is shipped with. If you do not explicitly reconfigure a parameter, it retains its default value.

- The “Memory Used” column displays the amount of memory used (in kilobytes) by the parameter at its current value. Some related parameters draw from the same memory pool. For instance, the memory used for stack size and stack guard size is already accounted for in the memory used for number of user connections. If you added the memory used by each of these parameters separately, it would total more than the amount actually used. In the “Memory Used” column, parameters that “share” memory with other parameters are marked with a hash mark (“#”).
- The “Config Value” column displays the most recent value to which the configuration parameter has been set. When you execute sp_configure to modify a dynamic parameter:
  - The configuration and run values are updated.
  - The configuration file is updated.
  - The change takes effect immediately.

  When you modify a static parameter:
  - The configuration value is updated.
  - The configuration file is updated.
  - The change takes effect only when you restart Adaptive Server.
- The “Run Value” column displays the value Adaptive Server is currently using. It changes when you modify a dynamic parameter’s value with sp_configure and, for static parameters, after you restart Adaptive Server.
- The “Unit” column displays the unit value in which the configuration parameter is displayed. Adaptive Server displays information in the following units:

Name of unit	**Unit description**
number	Displays the number of items a parameter is configured for.
clock ticks	Number of clock ticks a parameter is set for.
microseconds	Number of microseconds for which a parameter is set.
milliseconds	Number of milliseconds for which a parameter is set
seconds	Number of seconds for which a parameter is set
minutes	Number of minutes for which a parameter is set
hours	Number of hours for which a parameter is set
bytes	Number of bytes for which a parameter is set
days	Number of days for which a parameter is set
kilobytes	Number of kilobytes for which a parameter is set

Name of unit	Unit description
megabytes	Number of megabytes for which a parameter is set
memory pages (2K)	Number of 2K memory pages for which the parameter is set.
virtual pages (2K)	Number of 2K virtual pages for which the parameter is set.
logical pages	Number of logical pages for which the parameter is configured. This value depends on which logical page size your server is using; 2, 4, 8, or 16K.
percent	Displays the value of the configured parameter as a percentage.
ratio	Displays the value of the configured parameter as a ratio.
switch	Value of the parameter is either TRUE (the parameter is turned on or FALSE
id	ID of the configured parameter you are investigating.
name	Character string name assigned to the run or configure value of the parameter. For example, the string “binary” appears under the the run or configure value column for the output of `sp_configure "lock scheme"`.
row	Number of rows for which the specified parameter is configured.

- The “Type” column displays whether the configuration option is either static or dynamic. Changes to static parameters require that you restart Adaptive Server for the changes to take effect. Changes to Dynamic parameters take effect immediately without having to restart Adaptive Server.

The *sysconfigures* and *syscurconfigs* tables

The report displayed by sp_configure is constructed mainly from the master..sysconfigures and master..syscurconfigs system tables, with additional information coming from sysattributes, sysdevices, and other system tables.

The value column in the sysconfigures table records the last value set from sp_configure or the configuration file; the value column in syscurconfigs stores the value currently in use. For dynamic parameters, the two values match; for static parameters, which require a restart of the server to take effect, the two values are different if the values have been changed since Adaptive Server was last started. The values may also be different when the default values are used. In this case, sysconfigures stores 0, and syscurconfigs stores the value that Adaptive Server computes and uses.

sp_configure performs a join on sysconfigures and syscurconfigs to display the values reported by sp_configure.

Querying *syscurconfigs* and *sysconfigures*: an example

You might want to query sysconfigures and syscurconfigs to get information organized the way you want. For example, sp_configure without any arguments lists the memory used for configuration parameters, but it does not list minimum and maximum values. You can query these system tables to get a complete list of current memory usage, as well as minimum, maximum, and default values, with the following query:

```
select b.name, memory_used, minimum_value,
maximum_value, defvalue
from master.dbo.sysconfigures b,
master.dbo.syscurconfigs c
where b.config *= c.config and parent != 19
and b.config > 100
```

Details on configuration parameters

The following sections give both summary and detailed information about each of the configuration parameters. Parameters are listed by group; within each group, they are listed alphabetically.

In many cases, the maximum allowable values for configuration parameters are extremely high. The maximum value for your server is usually limited by available memory, rather than by sp_configure limitations.

Note To find the maximum supported values for your platform and version of Adaptive Server, see the table "Adaptive Server Specifications" in the *Installation Guide*.

Renamed configuration parameters

The following configuration parameters have been renamed:

Old name	New name	See
allow updates pre-11.0	allow updates to system tables	"allow updates to system tables" on page 151
calignment pre-11.0	memory alignment boundaries	"memory alignment boundary" on page 76
cclkrate pre-11.0	sql server clock tick length	"sql server clock tick length" on page 177
cfgcprot pre-11.0	permission cache entries	"permission cache entries" on page 199

Old name	New name	See
cguardsz pre-11.0	stack guard size	“stack guard size” on page 200
cindextrips pre-11.0	number of index trips	“number of index trips” on page 77
cmaxnetworks pre-11.0	max number network listeners	“max number network listeners” on page 129
cmaxscheds pre-11.0	i/o polling process count	“i/o polling process count” on page 164
cnalarm pre-11.0	number of alarms	“number of alarms” on page 168
cnblkio pre-11.0	disk i/o structures	“disk i/o structures” on page 86
cnmaxaio_engine pre-11.0	max async i/os per engine	“max async i/os per engine” on page 132
cnmaxaio_server pre-11.0	max async i/os per server	“max async i/os per server” on page 133
cnmbox pre-11.0	number of mailboxes	“number of mailboxes” on page 171
cnmsg pre-11.0	number of messages	“number of messages” on page 172
coamtrips pre-11.0	number of oam trips	“number of oam trips” on page 78
cpreallocext pre-11.0	number of pre-allocated extents	“number of pre-allocated extents” on page 172
cpu flush pre-11.0	cpu accounting flush interval	“cpu accounting flush interval” on page 152
cschedspins pre-11.0	runnable process search count	“runnable process search count” on page 175
csortbufsize pre-11.0	number of sort buffers	“number of sort buffers” on page 173
ctimemax pre-11.0	cpu grace time	“cpu grace time” on page 153
database size pre-11.0	default database size	“default database size” on page 154
default language pre-11.0	default language id	“default language id” on page 106
devices pre-11.0	number of devices	“number of devices” on page 86
fillfactor pre-11.0	default fill factor percent	“default fill factor percent” on page 154
i/o flush pre-11.0	i/o accounting flush interval	“i/o accounting flush interval” on page 163
locks pre-11.0	number of locks	“number of locks” on page 109
lock promotion HWM	page lock promotion HWM	“page lock promotion HWM” on page 165
lock promotion LWM	page lock promotion LWM	“page lock promotion LWM” on page 166
lock promotion PCT	page lock promotion PCT	“page lock promotion PCT” on page 167
maximum network packet size pre-11.0	max network packet size	“max network packet size” on page 126
mrstart pre-11.0	shared memory starting address	“shared memory starting address” on page 135
nested trigger pre-11.0	allow nested triggers	“allow nested triggers” on page 150
open databases pre-11.0	number of open databases	“number of open databases” on page 117
open objects pre-11.0	number of open objects	“number of open objects” on page 120
password expiration interval pre-11.0	systemwide password expiration	“systemwide password expiration” on page 192
pre-read packets pre-11.0	remote server pre-read packets	“remote server pre-read packets” on page 130
procedure cache percent pre-12.5	procedure cache size	“procedure cache size” on page 79

Old name	New name	See
recovery information pre-11.0	print recovery information	"print recovery information" on page 71
recovery interval pre-11.0	recovery interval in minutes	"recovery interval in minutes" on page 72
remote access pre-11.0	allow remote access	"allow remote access" on page 124
remote connections pre-11.0	number of remote connections	"number of remote connections" on page 129
remote logins pre-11.0	number of remote logins	"number of remote logins" on page 129
remote sites pre-11.0	number of remote sites	"number of remote sites" on page 130
sql server code size pre-11.0	executable codesize + overhead	"executable codesize + overhead" on page 116
T 1204 pre-11.0	print deadlock information	"print deadlock information" on page 174
T 1603 pre-11.0	allow sql server async i/o	"allow sql server async i/o" on page 85
T 1610 pre-11.0	tcp no delay	"tcp no delay" on page 131
T 1611 pre-11.0	lock shared memory	"lock shared memory" on page 143
tape retention pre-11.0	tape retention in days	"tape retention in days" on page 75
total memory pre-12.5	total logical memory	"total logical memory" on page 145
user connections pre-11.0	number of user connections	"number of user connections" on page 197

Replaced configuration parameter

The new lock spinlock ratio parameter replaces the page lock spinlock ration configuration parameter.

New configuration parameter

Summary information	
Default value	0
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The new number of java sockets parameter is necessary to enable the Java VM and the java.net classes Sybase supports. To open 10 sockets, for example, enter

```
sp_configure "number of java sockets", 10
```

Backup and recovery

The following parameters configure Adaptive for backing up and recovering data:

number of large i/o buffers

Summary information	
Default value	6
Valid values	1–256
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The number of large i/o buffers parameter sets the number of allocation unit-sized buffers reserved for performing large I/O for certain Adaptive Server utilities. These large I/O buffers are used primarily by the load database command. load database uses one buffer to load the database, regardless of the number of stripes it specifies. load database then uses up to 32 buffers to clear the pages for the database it is loading. These buffers are not used by load transaction. If you need to perform more than six load database commands concurrently, configure one large I/O buffer for each load database command.

create database and alter database use these buffers for large I/O while clearing database pages. Each instance of create database or load database can use up to 32 large I/O buffers.

These buffers are also used by disk mirroring and by some dbcc commands.

Note In Adaptive Server version 12.5.03 and above, the size of the large I/O buffers is now one allocation (256 pp), not one extent (8 pp). The server thus requires more memory allocation for large buffers. For example, a disk buffer that required memory for 8 pages in earlier versions now requires memory for 256 pages.

print recovery information

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)

Summary information	
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The print recovery information parameter determines what information Adaptive Server displays on the console during recovery. (Recovery is done on each database at Adaptive Server start-up and when a database dump is loaded.) The default value is 0, which means that Adaptive Server displays only the database name and a message saying that recovery is in progress. The other value is 1, which means that Adaptive Server displays information about each individual transaction processed during recovery, including whether it was aborted or committed.

recovery interval in minutes

Summary information	
Default value	5
Range of values	1–32767
Status	Dynamic
Display level	Basic
Required role	System Administrator

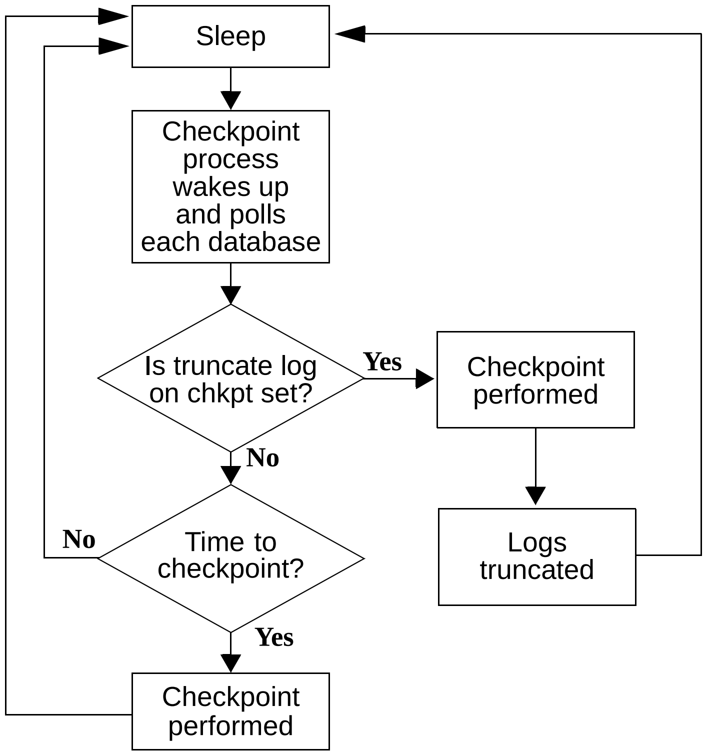
The recovery interval in minutes parameter sets the maximum number of minutes per database that Adaptive Server uses to complete its recovery procedures in case of a system failure. The recovery procedure rolls transactions backward or forward, starting from the transaction that the checkpoint process indicates as the oldest active transaction. The recovery process has more or less work to do depending on the value of recovery interval in minutes.

Adaptive Server estimates that 6000 rows in the transaction log require 1 minute of recovery time. However, different types of log records can take more or less time to recover. If you set recovery interval in minutes to 3, the checkpoint process writes changed pages to disk only when syslogs contains more than 18,000 rows since the last checkpoint.

Note The recovery interval has no effect on long-running, minimally logged transactions (such as create index) that are active at the time Adaptive Server fails. It may take as much time to reverse these transactions as it took to run them. To avoid lengthy delays, dump each database after index maintenance operations.

Adaptive Server uses the recovery interval in minutes setting and the amount of activity on each database to decide when to checkpoint each database. When Adaptive Server checkpoints a database, it writes all **dirty pages** (data pages in cache that have been modified) to disk. This may create a brief period of high I/O, called a *checkpoint spike*.The checkpoint also performs a few other maintenance tasks, including truncating the transaction log for each database for which the truncate log on chkpt option has been set. About once per minute, the sleeping checkpoint process "wakes up," checks the truncate log on chkpt setting, and checks the recovery interval to determine if a checkpoint is needed. Figure 4-1 shows the logic used by Adaptive Server during this process.

Figure 4-1: The checkpoint process



You may want to change the recovery interval if your application and its use change. For example, you may want to shorten the recovery interval when there is an increase in update activity on Adaptive Server. Shortening the recovery interval causes more frequent checkpoints, with smaller, more frequent checkpoint spikes, and slows the system slightly. On the other hand, setting the recovery interval too high might cause the recovery time to be unacceptably long. The spikes caused by checkpointing can be reduced by reconfiguring the housekeeper free write percent parameter. See “housekeeper free write percent” on page 158 for further information. For more information on the performance implications of recovery interval in minutes, see “Memory Use and Performance” in the *Performance and Tuning: Basics*.

Use sp_sysmon to determine how a particular recovery interval affects the system. See the *Performance and Tuning Guide* for more information.

tape retention in days

Summary information	
Default value	0
Range of values	0–365
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The tape retention in days parameter specifies the number of days you intend to retain each tape after it has been used for either a database or a transaction log dump. It is intended to keep you from accidentally overwriting a dump tape.

For example, if you have set tape retention in days to 7 days, and you try to use the tape before 7 days have elapsed since the last time you dumped to that tape, Backup Server issues a warning message.

You can override the warning by using the with init option when executing the dump command. Doing this will cause the tape to be overwritten and all data on the tape to be lost.

Both the dump database and dump transaction commands provide a retaindays option, which overrides the tape retention in days value for a particular dump. See "Protecting dump files from being overwritten" on page 911 for more information.

Cache manager

The parameters in this group configure the data and procedure caches.

global async prefetch limit

Summary information	
Default value	10
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The global async prefetch limit parameter specifies the percentage of a buffer pool that can hold the pages brought in by asynchronous prefetch that have not yet been read. This parameter sets the limit for all pools in all caches for which the limit has not been set explicitly with sp_poolconfig.

If the limit for a pool is exceeded, asynchronous prefetch is temporarily disabled until the percentage of unread pages falls below the limit. For more information, see “Tuning Asynchronous Prefetch” in the *Performance and Tuning Guide: Optimizer and Abstract Plans*.

global cache partition number

Summary information	
Default value	1
Range of values	1-64, as powers of 2
Status	Static
Display level	Intermediate
Required role	System Administrator

global cache partition number sets the default number of cache partitions for all data caches. The number of partitions for a particular cache can be set with sp_cacheconfig; the local value takes precedence over the global value.

Use cache partitioning to reduce cache spinlock contention; in general, if spinlock contention exceeds 10 percent, partitioning the cache should improve performance. Doubling the number of partitions cuts spinlock contention by about one-half.

See “Adding cache partitions” on page 647 for information on configuring cache partitions. See “Tuning Asynchronous Prefetch” in the *Performance and Tuning Guide: Optimizer and Abstract Plans* for information.

memory alignment boundary

Summary information	
Default value	Logical page size
Range of values	2048[a] – 16384 a. Minimum determined by server’s logical page size
Status	Static
Display level	Comprehensive
Required role	System Administrator

The memory alignment boundary parameter determines the memory address boundary on which data caches are aligned.

Some machines perform I/O more efficiently when structures are aligned on a particular memory address boundary. To preserve this alignment, values for memory alignment boundary should always be powers of two between the logical page size and 2048K.

Note The memory alignment boundary parameter is included for support of certain hardware platforms. Do not modify it unless you are instructed to do so by Sybase Technical Support.

number of index trips

Summary information	
Default value	0
Range of values	0–65535
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The number of index trips parameter specifies the number of times an aged index page traverses the most recently used/least recently used (MRU/LRU) chain before it is considered for swapping out. As you increase the value of number of index trips, index pages stay in cache for longer periods of time.

A data cache is implemented as an MRU/LRU chain. As the user threads access data and index pages, these pages are placed on the MRU end of the cache’s MRU/LRU chain. In some high transaction environments (and in some benchmarks), it is desirable to keep index pages in cache, since they will probably be needed again soon. Setting number of index trips higher keeps index pages in cache longer; setting it lower allows index pages to be swapped out of cache sooner.

You do not need to set the number of index pages parameter for relaxed LRU pages. For more information, see Chapter 19, “Configuring Data Caches.”

Note If the cache used by an index is relatively small (especially if it shares space with other objects) and you have a high transaction volume, do not set number of index trips too high. The cache can flood with pages that do not age out, and this may lead to the timing out of processes that are waiting for cache space.

number of oam trips

Summary information	
Default value	0
Range of values	0–65535
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The number of oam trips parameter specifies the number of times an *object allocation map* (OAM) page traverses the MRU/LRU chain before it is considered for swapping out. The higher the value of number of oam trips, the longer aged OAM pages stay in cache.

Each table, and each index on a table, has an OAM page. The OAM page holds information on pages allocated to the table or index and is checked when a new page is needed for the index or table. (See “page utilization percent” on page 87 for further information.) A single OAM page can hold allocation mapping for between 2,000 and 63,750 data or index pages.

The OAM pages point to the allocation page for each allocation unit where the object uses space. The allocation pages, in turn, track the information about extent and page usage within the allocation unit.

In some environments and benchmarks that involve significant allocations of space (that is, massive bulk copy operations), keeping OAM pages in cache longer improves performance. Setting number of oam trips higher keeps OAM pages in cache.

Note If the cache is relatively small and used by a large number of objects, do not set number of oam trips too high. This may result in the cache being flooded with OAM pages that do not age out, and user threads may begin to time out.

procedure cache size

Summary information	
Default value	3271
Range of values	3271 – 2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

Specifies the size of the procedure cache in 2K pages. Adaptive Server uses the procedure cache while running stored procedures. If the server finds a copy of a procedure already in the cache, it does not need to read it from the disk. Adaptive Server also uses space in the procedure cache to compile queries while creating stored procedures.

Since the optimum value for procedure cache size differs from application to application, resetting it may improve Adaptive Server's performance. For example, if you run many different procedures or ad hoc queries, your application uses the procedure cache more heavily, so you may want to increase this value.

Warning! If procedure cache size is too small, Adaptive Server's performance will be greatly affected.

If you are upgrading

If you are upgrading, procedure cache size is set to the size of the original procedure cache at the time of upgrade. procedure cache size is dynamically configurable, subject to the amount of max memory currently configured.

total data cache size

Summary information	
Default value	0
Range of values	0 – 2147483647
Status	Calculated
Display level	Basic
Required role	System Administrator

The total data cache size parameter reports the amount of memory, in kilobytes, that is currently available for data, index, and log pages. It is a calculated value that is not directly user-configurable.

The amount of memory available for the data cache can be affected by a number of factors, including:

- The amount of physical memory available on your machine
- The values to which the following parameters are set:
    - total logical memory
    - number of user connections
    - total procedure cache percent
    - number of open databases
    - number of open objects
    - number of open indexes
    - number of devices

A number of other parameters also affect the amount of available memory, but to a lesser extent.

For information on how Adaptive Server allocates memory and for information on data caches, see “Details on configuration parameters” on page 68.

Component Integration Services administration

The following parameters configure Adaptive Server for Component Integration Services.

cis bulk insert array size

Summary information	
Default value	50
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

When performing a bulk transfer of data from Adaptive Server Enterprise to another Adaptive Server Enterprise, CIS buffers rows internally, and asks the Open Client bulk library to transfer them as a block. The size of the array is controlled by cis bulk insert array size. The default is 50 rows, and the property is dynamic, allowing it to be changed without server reboot

cis bulk insert batch size

Summary information	
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The cis bulk insert batch size parameter determines how many rows from the source table(s) are to be bulk copied into the target table as a single batch using select into.

If the parameter is left at zero (the default), all rows are copied as a single batch. Otherwise, after the count of rows specified by this parameter has been copied to the target table, the server issues a bulk commit to the target server, causing the batch to be committed.

If a normal client-generated bulk copy operation (such as that produced by the bcp utility) is received, then the client is expected to control the size of the bulk batch, and the server ignores the value of this configuration parameter.

cis cursor rows

Summary information	
Default value	50
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The cis cursor rows parameter specifies the cursor row count for cursor open and cursor fetch operations. Increasing this value means more rows will be fetched in one operation. This increases speed but requires more memory. The default is 50.

cis packet size

Summary information	
Default value	512
Range of values	512–32768
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The cis packet size parameter specifies the size of Tabular Data Stream™ (TDS) packets that are exchanged between the server and a remote server when a connection is initiated.

The default packet size on most systems is 512 bytes, and this may be adequate for most applications. However, larger packet sizes may result in significantly improved query performance, especially when text and image or bulk data is involved.

If a packet size larger than the default is specified, and the requested server is a System 10 or later Adaptive Server, then the target server must be configured to allow variable-length packet sizes. Adaptive Server configuration parameters of interest in this case are:

- additional netmem
- maximum network packet size

cis rpc handling

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The cis rpc handling parameter specifies the default method for remote procedural call (RPC) handling. Setting cis rpc handling to 0 sets the Adaptive Server site handler as the default RPC handling mechanism. Setting the parameter to 1 forces RPC handling to use Component Integration Service access methods. For more information, see the discussion on set cis rpc handling in the *Component Integration Services User's Guide*.

enable cis

Summary information	
Default value	1
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The enable cis parameter enables or disables Component Integration Service.

enable file access

Summary information	
Default value	1
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

Enables access through proxy tables to the External File System. Requires a license for ASE_XFS.

enable full-text search

Summary information	
Default value	1
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

Enables Enhances Full-Text Search services. Requires a license for ASE_EFTS.

max cis remote connections

Summary information	
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

The max cis remote connections parameter specifies the maximum number of concurrent Client-Library connections that can be made to remote servers by Component Integration Services.

By default, Component Integration Services allows up to four connections per user to be made simultaneously to remote servers. If you set the maximum number of users to 25, up to 100 simultaneous Client-Library connections would be allowed by Component Integration Services.

If this number does not meet the needs of your installation, you can override the setting by specifying exactly how many outgoing Client-Library connections you want the server to be able to make at one time.

Disk I/O

The parameters in this group configure Adaptive Server’s disk I/O.

allow sql server async i/o

Summary information	
Default value	1
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The allow sql server async i/o parameter enables Adaptive Server to run with asynchronous disk I/O. Use asynchronous disk I/O, you have to enable it on *both* Adaptive Server *and* your operating system. See your operating system documentation for information on enabling asynchronous I/O at the operating system level.

In all circumstances, disk I/O runs faster asynchronously than synchronously. This is because when Adaptive Server issues an asynchronous I/O, it does not have to wait for a response before issuing further I/Os.

disable disk mirroring

Summary information	
Default value	1
Valid values	1, 0
Status	Static
Display level	Comprehensive
Required role	System Administrator

disable disk mirroring enables or disables disk mirroring for Adaptive Server. This is a global variable; Adaptive Server does not perform any disk mirroring after this configuration parameter is set to 1 and Adaptive Server is restarted. Setting disable disk mirroring to 0 enables disk mirroring.

Note Disk mirroring must be disabled if you configure Adaptive Server for Failover in a high availability system.

disk i/o structures

Summary information	
Default value	256
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The disk i/o structures parameter specifies the initial number of disk I/O control blocks Adaptive Server allocates at start-up.

User processes require a disk I/O control block before Adaptive Server can initiate an I/O request for the process. The memory for disk I/O control blocks is preallocated when Adaptive Server starts. You should configure disk i/o structures to as high a value as your operating system allows, to minimize the chance of running out of disk I/O structures. Refer to your operating system documentation for information on concurrent disk I/Os.

Use sp_sysmon to determine whether you need to allocate more disk I/O structures. See the *Performance and Tuning Guide*. You can set the max asynch i/os per server configuration parameter to the same value as disk i/o structures. See “max async i/os per server” on page 133 for more information.

number of devices

Summary information	
Default value	10
Range of values	1–256
Status	Dynamic
Display level	Basic
Required role	System Administrator

The number of devices parameter controls the number of database devices Adaptive Server can use. It does not include devices used for database or transaction log dumps.

When you execute disk init, you can also assign the device number (the vdevno). although this value is optional. If you do not assign a vdevno, Adaptive Server assigns the next available virtual device number.

If you do assign a virtual device number, each device number must be unique among the device numbers used by Adaptive Server. The number 0 is reserved for the master device. Legal numbers are 1–256. However, the highest number must be 1 less than the number of database devices you have configured for Adaptive Server. For example, if you configured your server for 10 devices, the legal range of device numbers is 1–9.

To determine which numbers are currently in use, run sp_helpdevice and look in the device_number column of output.

If you want to lower the number of devices value after you have added database devices, you must first check to see what device numbers are already in use by database devices. The following command prints the highest value in use:

```
select max(low/power(2,24))+1
    from master..sysdevices
```

Warning! If you set the number of devices value too low in your configuration file, Adaptive Server cannot start. You can find the devices in use by checking the sysdevices system table.

---

page utilization percent

Summary information	
Default value	95
Range of values	1–100
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The page utilization percent parameter is used during page allocations to control whether Adaptive Server scans a table’s OAM (*object allocation map*) to find unused pages or simply allocates a new extent to the table. (See “number of oam trips” on page 78 for more information on the OAM.) The page utilization percent parameter is a performance optimization for servers with very large tables; it reduces the time needed to add new space.

If page utilization percent is set to 100, Adaptive Server scans through all OAM pages to find unused pages allocated to the object before allocating a new extent. When this parameter is set lower than 100, Adaptive Server compares the page utilization percent setting to the ratio of used and unused pages allocated to the table, as follows:

```
100 * used pages/(used pages + unused pages)
```

If the page utilization percent setting is lower than the ratio, Adaptive Server allocates a new extent instead of searching for the unused pages.

For example, when inserting data into a 10GB table that has 120 OAM pages and only 1 unused data page:

- A page utilization percent of 100 tells Adaptive Server to scan through all 120 OAM pages to locate an unused data page.
- A page utilization percent of 95 allows Adaptive Server to allocate a new extent to the object, because 95 is lower than the ratio of used pages to used and unused pages.

A low page utilization percent value results in more unused pages. A high page utilization percent value slows page allocations in very large tables, as Adaptive Server performs an OAM scan to locate each unused page before allocating a new extent. This increases logical and physical I/O.

If page allocations (especially in the case of large inserts) seem to be slow, you can lower the value of page utilization percent, but be sure to reset it after inserting the data. A lower setting affects all tables on the server and results in unused pages in all tables.

Fast bulk copy ignores the page utilization percent setting and always allocates new extents until there are no more extents available in the database.

DTM administration

The following parameters configure distributed transaction management (DTM) facilities:

dtm detach timeout period

Summary information	
Default value	0 (minutes)
Valid values	0 to 2147483647 (minutes)
Status	Dynamic
Display level	10
Required role	System Administrator

dtm detach timeout period sets the amount of time, in minutes, that a distributed transaction branch can remain in the detached state. In some X/Open XA environments, a transaction may become detached from its thread of control (usually to become attached to a different thread of control). Adaptive Server permits transactions to remain in a detached state for the length of time specified by dtm detach timeout period. After this time has passed, Adaptive Server rolls back the detached transaction.

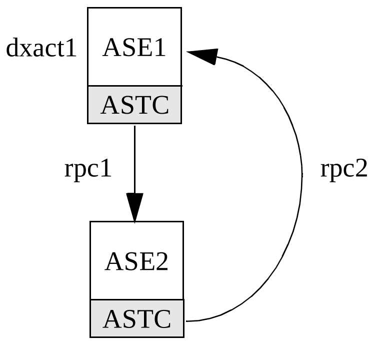
dtm lock timeout period

Summary information	
Default value	300 (seconds)
Valid values	1 to 2147483647 (seconds)
Status	Dynamic
Display level	10
Required role	System Administrator

dtm lock timeout period sets the maximum amount of time, in seconds, that a distributed transaction branch will wait for lock resources to become available. After this time has passed, Adaptive Server considers the transaction to be in a deadlock situation, and rolls back the transaction branch that triggered the deadlock. This ultimately rolls back the entire distributed transaction.

Distributed transactions may potentially deadlock themselves if they propagate a transaction to a remote server, and in turn, the remote server propagates a transaction back to the originating server. This situation is shown in Figure 4-2. In Figure 4-2, the work of distributed transaction “dxact1” is propagated to Adaptive Server 2 via “rpc1.” Adaptive Server 2 then propagates the transaction back to the coordinating server via “rpc2.” “rpc2” and “dxact1” share the same gtrid but have different branch qualifiers, so they cannot share the same transaction resources. If “rpc2” is awaiting a lock held by “dxact1”, then a deadlock situation exists.

Figure 4-2: Distributed transaction deadlock



Adaptive Server does not attempt to detect inter-server deadlocks. Instead, it relies on dtm lock timeout period. In Figure 4-2, after dtm lock timeout period has expired, the transaction created for "rpc2" is aborted. This causes Adaptive Server 2 to report a failure in its work, and "dxact1" is ultimately aborted as well.

The value of dtm lock timeout period applies only to distributed transactions. Local transactions may use a lock timeout period with the server-wide lock wait period parameter.

Note Adaptive Server does not use dtm lock timeout period to detect deadlocks on system tables.

enable DTM

Summary information	
Default value	0 (off)
Valid values	0 (off), 1(on)
Status	Static
Display level	10
Required role	System Administrator

enable DTM enables or disables the Adaptive Server Distributed Transaction Management (DTM) feature. When the DTM feature is enabled, you can use Adaptive Server as a resource manager in X/Open XA and MSDTC systems. You must reboot the server for this parameter to take effect. See the *XA Interface Integration Guide for CICS, Encina, and TUXEDO* for more information about using Adaptive Server in an X/Open XA environment. See *Using Adaptive Server Distributed Transaction Management Features* for information about transactions in MSDTC environments, and for information about Adaptive Server native transaction coordination services.

Note The license information and the Run value for enable DTM are independent of each other. Whether or not you have a license for DTM, the Run value and the Config value are set to 1 after you reboot Adaptive Server. And until you have a license, you cannot run DTM. If you have not installed a valid license, Adaptive Server logs an error message and does not activate the feature. See your Installation Guide for information about installing license keys.

The license information and the configuration value are independent of each other.

enable xact coordination

Summary information	
Default value	1 (on)
Valid values	0 (off), 1(on)
Status	Static
Display level	10
Required role	System Administrator

enable xact coordination enables or disables Adaptive Server transaction coordination services. When this parameter is set to 1 (on), coordination services are enabled, and the server can propagate transactions to other Adaptive Servers. This may occur when a transaction executes a remote procedure call (RPC) to update data in another server, or updates data in another server using Component Integration Services (CIS). Transaction coordination services ensure that updates to remote Adaptive Server data commit or roll back with the original transaction.

If this parameter is set to 0 (off), Adaptive Server will not coordinate the work of remote servers. Transactions can still execute RPCs and update data using CIS, but Adaptive Server cannot ensure that remote transactions are rolled back with the original transaction or that remote work is committed along with an original transaction, if remote servers experience a system failure. This corresponds to the behavior of Adaptive Server versions prior to version 12.x.

number of dtx participants

Summary information	
Default value	500
Valid values	100 to 2147483647
Status	Dynamic
Display level	10
Required role	System Administrator

number of dtx participants sets the total number of remote transactions that the Adaptive Server transaction coordination service can propagate and coordinate at one time. A DTX participant is an internal memory structure that the coordination service uses to manage a remote transaction branch. As transactions are propagated to remote servers, the coordination service must obtain new DTX participants to manage those branches.

By default, Adaptive Server can coordinate 500 remote transactions. Setting number of dtx participants to a smaller number reduces the number of remote transactions that the server can manage. If no DTX participants are available, new distributed transactions will be unable to start. In-progress distributed transactions may abort if no DTX participants are available to propagate a new remote transaction.

Setting number of dtx participants to a larger number increases the number of remote transaction branches that Adaptive Server can handle, but also consumes more memory.

Optimizing the number of dtx participants for your system

During a peak period, use sp_monitorconfig to examine the use of DTX participants:

```
sp_monitorconfig "number of dtx participants"
```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name             num_free num_active   pct_act    Max_Used   Reused
--------------   -------- ----------  ---------   --------   ------
number of dtx          80         20      4.00         210     NA
```

If the #Free value is zero or very low, new distributed transactions may be unable to start due to a lack of DTX participants. Consider increasing the number of dtx participants value.

If the #Max Ever Used value is too low, unused DTX participants may be consuming memory that could be used by other server functions. Consider reducing the value of number of dtx participants.

strict dtm enforcement

Summary information	
Default value	0 (off)
Valid values	0 (off), 1(on)
Status	Static
Display level	10
Required role	System Administrator

strict dtm enforcement determines whether or not Adaptive Server transaction coordination services will strictly enforce the ACID properties of distributed transactions.

In environments where Adaptive Server should propagate and coordinate transactions only to other Adaptive Servers that support transaction coordination, set strict dtm enforcement to 1 (on). This ensures that transactions are propagated only to servers that can participate in Adaptive Server-coordinated transactions, and transactions complete in a consistent manner. If a transaction attempts to update data in a server that does not support transaction coordination services, Adaptive Server aborts the transaction.

In heterogeneous environments, you may want to make use of servers that do not support transaction coordination. This includes older versions of Adaptive Server and non-Sybase database stores configured using CIS. Under these circumstances, you can set strict dtm enforcement to 0 (off). This allows Adaptive Server to propagate transactions to legacy Adaptive Servers and other data stores, but does not ensure that the remote work of these servers is rolled back or committed with the original transaction.

txn to pss ratio

Summary information	
Default value	16
Valid values	1 to 2147483647
Status	Static
Display level	1
Required role	System Administrator

Adaptive Server manages transactions as configurable server resources. Each time a new transaction begins, Adaptive Server must obtain a free **transaction descriptor** from a global pool that is created at boot time. Transaction descriptors are internal memory structures that Adaptive Server uses to represent active transactions.

Adaptive Server requires one free transaction descriptor for:

- The outer block of each server transaction. The outer block of a transaction may be created explicitly when a client executes a new begin transaction command. Adaptive Server may also implicitly create an outer transaction block when clients use Transact-SQL to modify data without using begin transaction to define the transaction.

  Note Subsequent, nested transaction blocks, created with additional begin transaction commands, do not require additional transaction descriptors.

- Each database accessed in a **multi-database transaction**. Adaptive Server must obtain a new transaction descriptor each time a transaction uses or modifies data in a new database.

txn to pss ratio determines the total number of transaction descriptors available to the server. At start-up, this ratio is multiplied by the number of PSS structures to create the transaction descriptor pool:

```
# of transaction descriptors = PSS structures * txn to pss ratio
```

The default value, 16, ensures compatibility with earlier versions of Adaptive Server. Prior to version 12.x, Adaptive Server allocated 16 transaction descriptors for each user connection. In version 12.x and later, the number of simultaneous transactions is limited only by the number of transaction descriptors available in the server.

Note You can have as many databases in a user transaction as there are in your Adaptive Server installation. For example, if your Adaptive Server has 25 databases, you can include 25 databases in your user transactions.

Optimizing the txn to pss ratio for your system

During a peak period, use sp_monitorconfig to examine the use of transaction descriptors:

```
    sp_monitorconfig "txn to pss ratio"
```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name               num_free  num_active   pct_act    Max_Used   Reused
--------------     --------  ----------  ---------   --------   ------
txn to pss ratio   784       80          10.20       523        NA
```

If the #Free value is zero or very low, transactions may be delayed as Adaptive Server waits for transaction descriptors to become free in the server. In this case, you should consider increasing the value of txn to pss ratio.

If the #Max Ever Used value is too low, unused transaction descriptors may be consuming memory that can be used by other server functions. Consider reducing the value of txn to pss ratio.

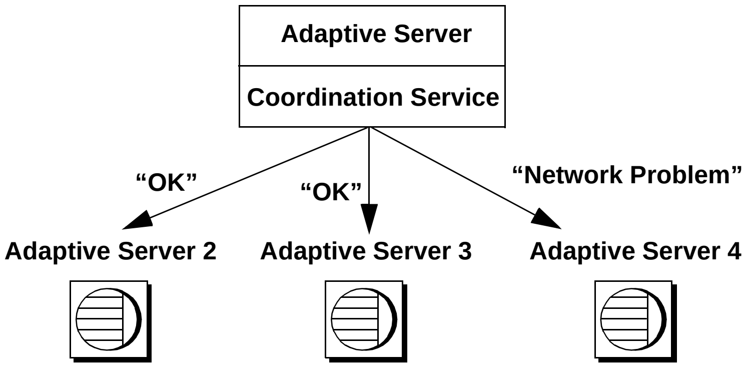
xact coordination interval

Summary information	
Default value	60 (seconds)
Valid values	1 to 2147483647 (seconds)
Status	Dynamic
Display level	10
Required role	System Administrator

xact coordination interval defines the length of time between attempts to resolve transaction branches that were propagated to remote servers.

The coordinating Adaptive Server makes regular attempts to resolve the work of remote servers participating in a distributed transaction. The coordinating server contacts each remote server participating in the distributed transaction in a serial manner, as shown in Figure 4-3. The coordination service may be unable to resolve a transaction branch for a variety of reasons. For example, if the remote server is not reachable due to network problems, the coordinating server reattempts the connection after the time specified by xact coordination level.

Figure 4-3: Resolving remote transaction branches



With the default value of xact coordination interval, 60, Adaptive Server attempts to resolve remote transactions once every minute. Decreasing the value may speed the completion of distributed transactions, but only if the transactions are themselves resolved in less than a minute. Under normal circumstances, there is no performance penalty to decreasing the value of xact coordination interval.

Setting xact coordination interval to a higher number can slow the completion of distributed transactions, and cause transaction branches to hold resources longer than they normally would. Under normal circumstances, you should not increase the value of xact coordination interval beyond its default.

Error log

The parameters in this group configure the Adaptive Server error log and the logging of Adaptive Server events to the Windows NT Event Log.

event log computer name (Windows NT only)

Summary information	
Default value	'LocalSystem'
Valid values	• Name of an NT machine on the network configured to record Adaptive Server messages • 'LocalSystem' • 'NULL'
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The event log computer name parameter specifies the name of the Windows NT PC that logs Adaptive Server messages in its Windows NT Event Log. You can use this parameter to have Adaptive Server messages logged to a remote machine. This feature is available on Windows NT servers only.

A value of 'LocalSystem' or 'NULL' specifies the default local system.

You can also use the Server Config utility to set the event log computer name parameter by specifying the Event Log Computer Name under Event Logging.

Setting the event log computer name parameter with sp_configure or specifying the Event Log Computer Name under Event Logging overwrites the effects of the command line -G option, if it was specified. If Adaptive Server was started with the -G option, you can change the destination remote machine by setting the event log computer name parameter.

For more information about logging Adaptive Server messages to a remote site, see *Configuration Guide for Windows NT*.

event logging (Windows NT only)

Summary information	
Default value	1
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The event logging parameter enables and disables the logging of Adaptive Server messages in the Windows NT Event Log. This feature is available on Windows NT servers only.

The default value of 1 enables Adaptive Server message logging in the Windows NT Event Log; a value of 0 disables it.

You use the Server Config utility to set the event logging parameter by selecting "Use Windows NT Event Logging" under Event Logging.

Setting the event logging parameter or selecting "Use Windows NT Event Logging" overwrites the effects of the command line -g option, if it was specified.

log audit logon failure

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The log audit logon failure parameter specifies whether to log unsuccessful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.

A value of 1 requests logging of unsuccessful logins; a value of 0 specifies no logging.

log audit logon success

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The log audit logon success parameter specifies whether to log successful Adaptive Server logins to the Adaptive Server error log and, on Windows NT servers, to the Windows NT Event Log, if event logging is enabled.

A value of 1 requests logging of successful logins; a value of 0 specifies no logging.

Extended stored procedures

The parameters in this group affect the behavior of extended stored procedures (ESPs).

esp execution priority

Summary information	
Default value	8
Range of values	0–15
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The esp execution priority parameter sets the priority of the XP Server thread for ESP execution. ESPs can be CPU-intensive over long periods of time. Also, since XP Server resides on the same machine as Adaptive Server, XP Server can impact Adaptive Server's performance.

Use esp execution priority to set the priority of the XP Server thread for ESP execution. See the *Open Server Server-Library/C Reference Manual* for information about scheduling Open Server threads.

esp execution stacksize

Summary information	
Default value	34816
Range of values	34816–2^{14}
Status	Static
Display level	Comprehensive
Required role	System Administrator

The esp execution stacksize parameter sets the size of the stack, in bytes, to be allocated for ESP execution.

Use this parameter if you have your own ESP functions that require a larger stack size than the default, 34816.

esp unload dll

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The esp unload dll parameter specifies whether DLLs that support ESPs should be automatically unloaded from XP Server memory after the ESP call has completed.

If esp unload dll is set to 0, DLLs are not automatically unloaded. If it is set to 1, they are automatically unloaded.

If esp unload dll is set to 0, you can still unload individual DLLs explicitly at runtime, using sp_freedll.

start mail session (Windows NT only)

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The start mail session parameter enables and disables the automatic initiation of an Adaptive Server mail session when you start Adaptive Server. This feature is available on Windows NT servers only.

A value of 1 configures Adaptive Server to start a mail session the next time Adaptive Server is started. A value of 0 configures Adaptive Server not to start a mail session at the next restart.

If start mail session is 0, you can start an Adaptive Server mail session explicitly, using the xp_startmail system ESP.

Before setting the start mail session parameter, you must prepare your Windows NT system by creating a mailbox and mail profile for Adaptive Server. Then, you must create an Adaptive Server account for Sybmail. See *Configuration Guide for Windows NT* for information about preparing your system for Sybmail.

xp_cmdshell context

Summary information	
Default value	1
Valid values	0, 1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The xp_cmdshell context parameter sets the security context for the operating system command to be executed using the xp_cmdshell system ESP.

Setting xp_cmdshell context to 1 restricts the xp_cmdshell security context to users who have accounts at the operating system level. Its behavior is platform-specific. If xp_cmdshell context is set to 1, to use an xp_cmdshell ESP, an operating system user account must exist for the Adaptive Server user name. For example, an Adaptive Server user named “sa” will not be able to use xp_cmdshell unless he or she has an operating system level user account named “sa”.

On Windows NT, when xp_cmdshell context is set to 1, xp_cmdshell succeeds only if the user name of the user logging in to Adaptive Server is a valid Windows NT user name with Windows NT system administration privileges on the system on which Adaptive Server is running.

On other platforms, when xp_cmdshell context is set to 1, xp_cmdshell succeeds only if Adaptive Server was started by a user with “superuser” privileges at the operating system level. When Adaptive Server gets a request to execute xp_cmdshell, it checks the uid of the user name of the ESP requestor and runs the operating system command with the permissions of that uid.

If xp_cmdshell context is 0, the permissions of the operating system account under which Adaptive Server is running are the permissions used to execute an operating system command from xp_cmdshell. This allows users to execute operating commands that they would not ordinarily be able to execute under the security context of their own operating system accounts.

General information

The parameter in this group is not related to any particular area of Adaptive Server behavior.

configuration file

Summary information	
Default value	0
Range of values	N/A
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The configuration file parameter specifies the location of the configuration file currently in use. See "Using sp_configure with a configuration file" on page 57 for a complete description of configuration files.

In sp_configure output, the "Run Value" column displays only 10 characters. For this reason, the output may not display the entire path and name of your configuration file.

sampling percent

Summary information	
Default value	0
Range of values	1 to 99 percent
Status	Dynamic
Display level	Comprehensive
Required role	System or database administrator

sampling percent is the numeric value of the sampling percentage, such as 05 for 5%, 10 for 10%, and so on. The sampling integer is between zero (0) and one hundred (100)

To reduce I/O contention and resources, run update statistics using a sampling method, which can reduce the I/O and time when your maintenance window is small and the data set is large. If you are updating a large data set or table that is in constant use, being truncated and repopulated, you may want to do a statistical sampling to reduce the time and the size of the I/O.

You must use caution with sampling since the results are not fully accurate. Balance changes to histogram values against the savings in I/O.

Although a sampling of the data set may not be completely accurate, usually the histograms and density values are reasonable within an acceptable range.

When you are deciding whether or not to use sampling, consider the size of the data set, the time constraints you are working with, and if the histogram produced is as accurate as needed.

The percentage to use when sampling depends on your needs. Test various percentages until you receive a result that reflects the most accurate information on a particular data set.

Statistics are stored in the system tables systabstats and sysstatistics.

Java services

The parameters in this group enable and configure memory for Java in Adaptive Server. See the *Java in Adaptive Server Enterprise* manual for complete information about Java in the database.

If you use method calls to JDBC, you may need to increase the size of the execution stack available to the user. See "stack size" on page 202 for information about setting the stack size parameter.

enable java

Summary information	
Default value	0 (disabled)
Range of values	0 (disabled), 1 (enabled)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The enable java parameter enables and disables Java in the Adaptive Server database. You cannot install Java classes or perform any Java operations until the server is enabled for Java.

Note The license information and the Run value for enable java are independent of each other. Whether or not you have a license for java, the Run value and the Config value are set to 1 after you restart Adaptive Server. You cannot run Java until you have a license. If you have not installed a valid license, Adaptive Server logs an error message and does not activate the feature. See your installation guide for information about installing license keys.

enable enterprise java beans

Summary information	
Default value	0 (disabled)
Range of values	0 (disabled), 1 (enabled)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The enable enterprise java beans parameter enables and disables EJB Server in the Adaptive Server database. You cannot use EJB Server until the Adaptive Server is enabled for EJB Server.

Note The license information and the Run value for enable java beans are independent of each other. Whether or not you have a license for java, the Run value and the Config value are set to 1 after you restart Adaptive Server. You cannot run EJB Server until you have a license. If you have not installed a valid license, Adaptive Server logs an error message and does not activate the feature. See your installation guide for information about installing license keys.

size of global fixed heap

Summary information	
Default values	150 pages (32-bit version)
	300 pages (64-bit version)
Minimum values	10 pages (32-bit version)
	20 pages (64-bit version)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The size of global fixed heap parameter specifies the memory space for internal data structures and other needs.

If you change the size of the global fixed heap, you must also change the total logical memory by the same amount.

size of process object heap

Summary information	
Default values	1500 pages (32-bit version)
	3000 pages (64-bit version)
Minimum values	45 pages (32-bit version)
	90 pages (64-bit version)
Status	Dynamic
Display level	Basic
Required role	System Administrator

The size of process object fixed heap parameter specifies the total memory space for all processes using the Java VM.

If you change the size of process object fixed heap, you must change the total logical memory by that amount.

size of shared class heap

Summary information	
Default values	1536 pages (32-bit version)
	3072 pages (64-bit version)

Summary information	
Minimum values	650 pages (32-bit version) 1300 pages (64-bit version)
Status	Dynamic
Display level	Basic
Required role	System Administrator

The size of shared class heap parameter specifies the shared memory space for all Java classes called into the Java VM. Adaptive Server maintains the shared class heap server-wide for both user-defined and system-provided Java classes.

If you change the size of shared class heap, you must change the total logical memory by the same amount.

Languages

The parameters in this group configure languages, sort orders, and character sets.

default character set id

Summary information	
Default value	1
Range of values	0–255
Status	Static
Display level	Intermediate
Required role	System Administrator

The default character set id parameter specifies the number of the default character set used by the server. The default is set at installation time, and can be changed later with the Sybase installation utilities. See Chapter 7, “Configuring Character Sets, Sort Orders, and Languages,” for a discussion of how to change character sets and sort orders.

default language id

Summary information	
Default value	0

Summary information	
Range of values	0–32767
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The default language id parameter is the number of the language that is used to display system messages unless a user has chosen another language from those available on the server. us_english always has an ID of NULL. Additional languages are assigned unique numbers as they are added.

default sortorder id

Summary information	
Default value	50
Range of values	0–255
Status	Static
Display level	Comprehensive
Required role	System Administrator

The default sortorder id parameter is the number of the sort order that is installed as the default on the server. To change the default sort order, see Chapter 7, "Configuring Character Sets, Sort Orders, and Languages."

disable character set conversions

Summary information	
Default value	0 (enabled)
Valid values	0 (enabled), 1 (disabled)
Status	Static
Display level	Comprehensive
Required role	System Administrator

Changing disable character set conversions to 1 turns off character set conversion for data moving between clients and Adaptive Server. By default, Adaptive Server performs conversion on data moving to and from clients that use character sets that are different than the server's. For example, if some clients use Latin-1 (iso_1) and Adaptive Server uses Roman-8 (roman8) as its default character set, data from the clients is converted to Roman-8 when being loaded into Adaptive Server. For clients using Latin-1, the data is reconverted when it is sent to the client; for clients using the same character set as Adaptive Server, the data is not converted.

By setting disable character set conversions, you can request that no conversion take place. For example, if all clients are using a given character set, and you want Adaptive Server to store all data in that character set, you can set disable character set conversions to 1, and no conversion will take place.

Lock Manager

The parameters in this group configure locks.

lock address spinlock ratio

Summary information	
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the address lock spinlock ratio sets the number of rows in the internal address locks hash table that are protected by one spinlock.

Adaptive Server manages the acquiring and releasing of address locks using an internal hash table with 1031 rows (known as hash buckets). This table can use one or more spinlocks to serialize access between processes running on different engines.

Adaptive Server's default value for address lock spinlock ratio is 100, which defines 11 spinlocks for the address locks hash table. The first 10 spinlocks protect 100 rows each, and the eleventh spinlock protects the remaining 31 rows. If you specify a value of 1031 or greater for address lock spinlock ratio, Adaptive Server uses only 1 spinlock for the entire table.

number of locks

Summary information	
Default value	5000
Range of values	1000–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

The number of locks parameter sets the total number of available locks for all users on Adaptive Server.

The total number of locks needed by Adaptive Server depends on the number and nature of the queries that are running. The number of locks required by a query can vary widely, depending on the number of concurrent and parallel processes and the types of actions performed by the transactions. To see how many locks are in use at a particular time, use sp_lock.

For serial operation, we suggest that you can start with an arbitrary number of 20 locks for each active, concurrent connection.

Parallel execution requires more locks than serial execution. For example, if you find that queries use an average of five worker processes, try increasing, by one-third, the number of locks configured for serial operation.

If the system runs out of locks, Adaptive Server displays a server-level error message. If users report lock errors, it typically indicates that you need to increase number of locks; but remember that locks use memory. See "Number of locks" on page 611 for information.

Note Datarows locking may require that you change the value for number of locks. See the *Performance and Tuning Guide* for more information.

deadlock checking period

Summary information	
Default value	500
Range of values	0–2147483
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

deadlock checking period specifies the minimum amount of time (in milliseconds) before Adaptive Server initiates a deadlock check for a process that is waiting on a lock to be released. Deadlock checking is time-consuming overhead for applications that experience no deadlocks or very few, and the overhead grows as the percentage of lock requests that must wait for a lock also increases.

If you set this value to a nonzero value (*n*), Adaptive Server initiates a deadlock check after a process waits at least *n* milliseconds. For example, you can make a process wait at least 700 milliseconds for a lock before each deadlock check as follows:

```
sp_configure "deadlock checking period", 700
```

If you set this parameter to 0, Adaptive Server initiates deadlock checking when each process begins to wait for a lock. Any value less than the number of milliseconds in a clock tick is treated as 0. See “sql server clock tick length” on page 177 for more information.

Configuring deadlock checking period to a higher value produces longer delays before deadlocks are detected. However, since Adaptive Server grants most lock requests before this time elapses, the deadlock checking overhead is avoided for those lock requests. If your applications deadlock infrequently, set deadlock checking period to a higher value to avoid the overhead of deadlock checking for most processes. Otherwise, the default value of 500 should suffice.

Use sp_sysmon to determine the frequency of deadlocks in your system and the best setting for deadlock checking period. See the *Performance and Tuning Guide* for more information.

deadlock retries

Summary information	
Default value	5
Range of values	0–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

deadlock retries specifies the number of times a transaction can attempt to acquire a lock when deadlocking occurs during an index page split or shrink.

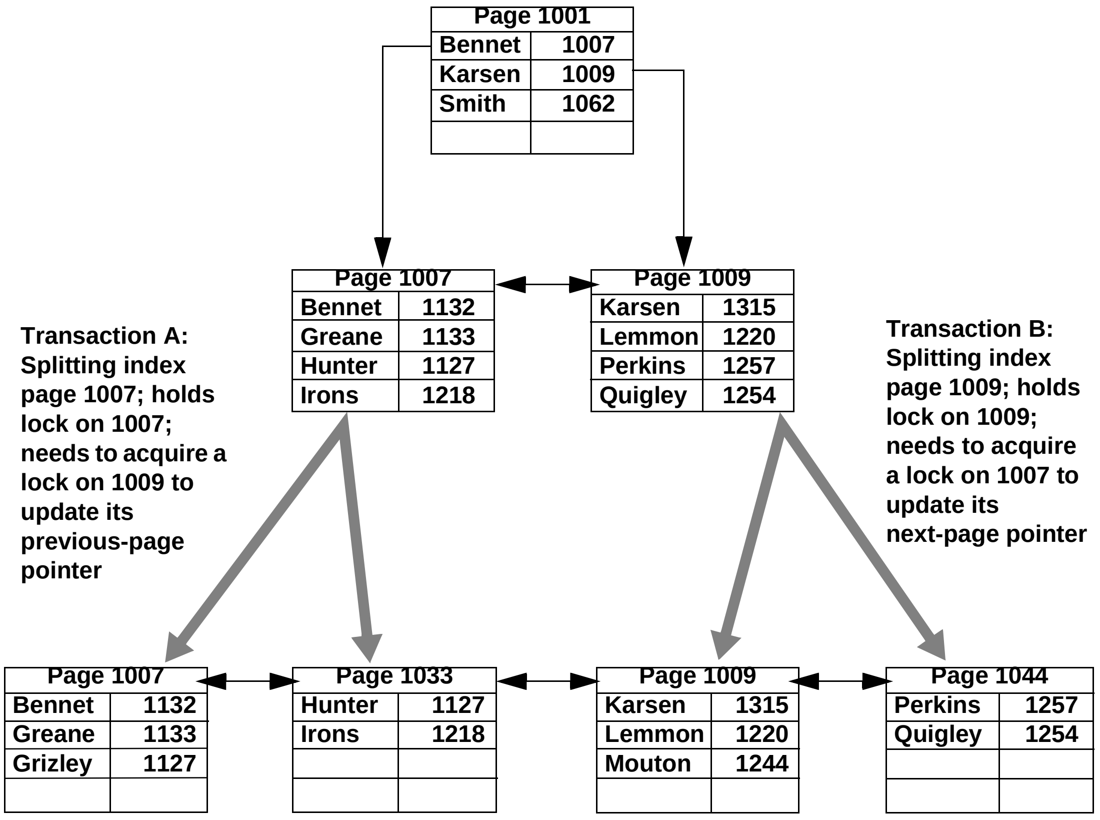
For example, Figure 4-4 illustrates the following scenario:

- Transaction A locks page 1007 and needs to acquire a lock on page 1009 to update the page pointers for a page split.
- Transaction B is also inserting an index row that causes a page split, holds a lock on page 1009, and needs to acquire a lock on page 1007.

In this situation, rather than immediately choosing a process as a deadlock victim, Adaptive Server relinquishes the index locks for one of the transactions. This often allows the other transaction to complete and release its locks.

For the transaction that surrendered its locking attempt, the index is rescanned from the root page, and the page split operation is attempted again, up to the number of times specified by deadlock retries.

Figure 4-4: Deadlocks during page splitting in a clustered index



sp_sysmon reports on deadlocks and retries. See the *Performance and Tuning Guide* for more information.

lock spinlock ratio

Summary information	
Default value	85
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

Adaptive Server manages the acquiring and releasing of locks using an internal hash table with a configurable number of hash buckets. On SMP systems, this hash table can use one or more spinlocks to serialize access between processes running on different engines. To set the number of hash buckets, use the lock hashtable size.

For Adaptive Servers running with multiple engines, the lock spinlock ratio sets a ratio that determines the number of lock hash buckets that are protected by one spinlock. If you increase lock hashtable size, the number of spinlocks increases, so the number of hash buckets protected by one spinlock remains the same.

Adaptive Server's default value for lock spinlock ratio is 85. With lock hashtable size set to the default value of 2048, the default spinlock ratio defines 26 spinlocks for the lock hash table. For more information about configuring spinlock ratios, see "Configuring spinlock ratio parameters" on page 668 of the *System Administration Guide*.

sp_sysmon reports on the average length of the hash chains in the lock hash table. See the *Performance and Tuning Guide* for more information.

lock hashtable size

Summary information	
Default value	2048
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

lock hashtable size specifies the number of *hash buckets* in the lock hash table. This table manages all row, page, and table locks and all lock requests. Each time a task acquires a lock, the lock is assigned to a hash bucket, and each lock request for that lock checks the same hash bucket. Setting this value too low results in large numbers of locks in each hash bucket and slows the searches. On Adaptive Servers with multiple engines, setting this value too low can also lead to increased spinlock contention. Do not set the value to less than the default value, 2048.

lock hashtable size must be a power of 2. If the value you specify is not a power of 2, sp_configure rounds the value to the next highest power of 2 and prints an informational message.

The optimal hash table size is a function of the number of distinct objects (pages, tables, and rows) that will be locked concurrently. The optimal hash table size is at least 20 percent of the number of distinct objects that need to be locked concurrently. See *Performance and Tuning Guide* for more information on configuring the lock hash table size.

lock scheme

Summary information	
Default value	allpages
Range of values	allpages, datapages, datarows
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

lock scheme sets the default locking scheme to be used by create table and select into commands when a lock scheme is not specified in the command.

The values for lock scheme are character data, so you must use 0 as a placeholder for the second parameter, which must be numeric, and specify allpages, datapages, or datarows as the third parameter:

```
sp_configure "lock scheme", 0, datapages
```

lock wait period

Summary information	
Default value	2147483647
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

lock wait period limits the number of seconds that tasks wait to acquire a lock on a table, data page, or data row. If the task does not acquire the lock within the specified time period, Adaptive Server returns error message 12205 to the user and rolls back the transaction.

The lock wait option of the set command sets a session-level number of seconds that a task will wait for a lock. It overrides the server-level setting for the session.

lock wait period, used with the session-level setting set lock wait nnn, is only applicable to user-defined tables. These settings have no influence on system table.

At the default value, all processes wait indefinitely for locks. To restore the default value, reset the value to 2147483647 or use:

```
sp_configure "lock wait period", 0, "default"
```

read committed with lock

Summary information	
Default value	0 (off)
Valid values	0 (off), 1(on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

read committed with lock determines whether an Adaptive Server using transaction isolation level 1 (read committed) holds shared locks on rows or pages of data-only-locked tables during select queries. For cursors, the option applies only to cursors declared as read-only. By default, this parameter is turned off to reduce lock contention and blocking. This parameter affects only queries on data-only locked tables.

For transaction isolation level 1, select queries on allpages-locked tables continue to hold locks on the page at the current position. Any updatable cursor on a data-only-locked table also holds locks on the current page or row. See the *Performance and Tuning Guide* for more information.

lock table spinlock ratio

Summary information	
Default value	20
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, the table lock spinlock ratio configuration parameter sets the number of rows in the internal table locks hash table that are protected by one **spinlock**.

Adaptive Server manages the acquiring and releasing of table locks using an internal hash table with 101 rows (known as hash buckets). This table can use one or more spinlocks to serialize access between processes running on different engines.

Adaptive Server's default value for table lock spinlock ratio is 20, which defines 6 spinlocks for the table locks hash table. The first 5 spinlocks protect 20 rows each; the sixth spinlock protects the last row. If you specify a value of 101 or greater for table lock spinlock ratio, Adaptive Server uses only 1 spinlock for the entire table.

Memory use

The following parameter optimizes Adaptive Server's memory use:

executable codesize + overhead

Summary information	
Default value	0
Range of values	0-2147483647
Status	Calculated
Display level	Basic
Required role	System Administrator

executable codesize + overhead reports the combined size (in kilobytes) of the Adaptive Server executable and overhead. It is a calculated value and is not user-configurable.

Metadata caches

The following parameters help set the metadata cache size for frequently used system catalog information. The *metadata cache* is a reserved area of memory used for tracking information on databases, indexes, or objects. The greater the number of open databases, indexes, or objects, the larger the metadata cache size. For a discussion of metadata caches in a memory-usage context, see "Open databases, open indexes, and open objects" on page 610.

number of open databases

Summary information	
Default value	12
Range of values	5–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

number of open databases sets the maximum number of databases that can be open simultaneously on Adaptive Server.

When you calculate a value, include the system databases master, model, sybsystemprocs, and tempdb. If you have installed auditing, include the sybsecurity database. Also, count the sample databases pubs2 and pubs3, the syntax database sybsyntax, and the dbcc database dbccdb if they are installed.

If you are planning to make a substantial change, such as loading a large database from another server, you can calculate an estimated metadata cache size by using sp_helpconfig. sp_helpconfig displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. A database metadata descriptor represents the state of the database while it is in use or cached between uses.

Optimizing the *number of open databases* parameter for your system

If Adaptive Server displays a message saying that you have exceeded the allowable number of open databases, you will need to adjust the value.

To set the number of open databases parameter optimally:

- Step 1: Determine the total number of databases (database metadata descriptors).
- Step 2: Reset number of open databases to that number.

- Step 3: Find the number of active databases (active metadata descriptors) during a peak period.
- Step 4: Reset number of open databases to that number, plus 10 percent.

The following section details the basic steps listed above.

1. Use the sp_countmetadata system procedure to find the total number of database metadata descriptors. For example:

   ```
   sp_countmetadata "open databases"
   ```

   The best time to run sp_countmetadata is when there is little activity on the server. Running sp_countmetadata during a peak time can cause contention with other processes.

   Suppose Adaptive Server reports the following information:

   ```
   There are 50 databases, requiring 1719 Kbytes of
   memory. The 'open databases' configuration parameter
   is currently set to 500.
   ```

2. Configure number of open databases with the value of 50:

   ```
   sp_configure "number of open databases", 50
   ```

   This new configuration is only a start; the ideal size should be based on the number of *active* metadata database cache descriptors, not the *total* number of databases.

3. During a peak period, find the number of active metadata descriptors. For example:

   ```
   sp_monitorconfig "open databases"
   ```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name               num_free   num_active   pct_act     Max_Used   Reused
--------------     --------    ---------   --------    --------   ------
number of open     50         20           40.00       26         No
```

   At this peak period, 20 metadata database descriptors are active; the maximum number of descriptors that have been active since the server was last started is 26.

   See sp_monitorconfig in the *Reference Manual: Procedures* for more information.

4. Configure number of open databases to 26, plus additional space for 10 percent more (about 3), for a total of 29:

   ```
   sp_configure "number of open databases", 29
   ```

If there is a lot of activity on the server, for example, if databases are being added or dropped, run sp_monitorconfig periodically. You will need to reset the cache size as the number of active descriptors changes.

number of open indexes

Summary information	
Default value	500
Range of values	100–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

number of open indexes sets the maximum number of indexes that can be used simultaneously on Adaptive Server.

If you are planning to make a substantial change, such as loading databases with a large number of indexes from another server, you can calculate an estimated metadata cache size by using sp_helpconfig. sp_helpconfig displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. An index metadata descriptor represents the state of an index while it is in use or cached between uses.

Optimizing the *number of open indexes* parameter for your system

The default run value is 500. If this number is insufficient, Adaptive Server displays a message after trying to reuse active index descriptors, and you will need to adjust this value.

In order to configure the number of open indexes parameter optimally, perform the following steps:

1 Use sp_countmetadata to find the total number of index metadata descriptors. For example:

```
sp_countmetadata "open indexes"
```

The best time to run sp_countmetadata is when there is little activity in the server. Running sp_countmetadata during a peak time can cause contention with other processes.

Suppose Adaptive Server reports the following information:

```
There are 698 user indexes in all database(s),
requiring 286.289 Kbytes of memory. The 'open
indexes' configuration parameter is currently set to
```

```
500.
```

2 Configure the number of open indexes parameter to 698 as follows:

```
sp_configure "number of open indexes", 698
```

This new configuration is only a start; the ideal size should be based on the number of *active* index metadata cache descriptors, not the total number of indexes.

3 During a peak period, find the number of active index metadata descriptors. For example:

```
sp_monitorconfig "open indexes"
```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name              num_free   num_active   pct_act     Max_Used   Reused
--------------    --------    ---------   --------    --------   ------
number of open     182         516        73.92       590        No
```

In this example, 590 is the maximum number of index descriptors that have been used since the server was last started.

See sp_monitorconfig in the *Reference Manual* for more information.

4 Configure the number of open indexes configuration parameter to 590, plus additional space for 10 percent more (59), for a total of 649:

```
sp_configure "number of open indexes", 649
```

If there is a lot of activity on the server, for example, if tables are being added or dropped, run sp_monitorconfig periodically. You will need to reset the cache size as the number of active descriptors changes.

number of open objects

Summary information	
Default value	500
Range of values	100–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

number of open objects sets the maximum number of objects that can be open simultaneously on Adaptive Server.

If you are planning to make a substantial change, such as loading databases with a large number of objects from another server, you can calculate an estimated metadata cache size by using sp_helpconfig. sp_helpconfig displays the amount of memory required for a given number of metadata descriptors, as well as the number of descriptors that can be accommodated by a given amount of memory. An object metadata descriptor represents the state of an object while it is in use, or cached between uses.

Optimizing the *number of open objects* parameter for your system

The default run value is 500. If this number is insufficient, Adaptive Server displays a message after trying to re-use active object descriptors. You will need to adjust this value.

To set the number of open objects parameter optimally:

1 Use sp_countmetadata to find the total number of object metadata cache descriptors. For example:

```
sp_countmetadata "open objects"
```

The best time to run sp_countmetadata is when there is little activity in the server. Running sp_countmetadata during a peak time can cause contention with other processes.

Suppose Adaptive Server reports the following information:

```
There are 340 user objects in all database(s),
requiring 140.781 Kbytes of memory. The 'open
objects' configuration parameter is currently set to
500
```

2 Configure the number of open objects parameter to that value, as follows:

```
sp_configure "number of open objects", 357
```

357 covers the 340 user objects, plus 5 percent to accommodate temporary tables.

This new configuration is only a start; the ideal size should be based on the number of *active* object metadata cache descriptors, not the *total* number of objects.

3 During a peak period, find the number of active metadata cache descriptors, for example:

```
sp_monitorconfig "open objects"
```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name              num_free   num_active   pct_act     Max_Used   Reused
--------------    --------    ---------   --------    --------   ------
number of open     160         357        71.40       397        No
```

In this example, 397 is the maximum number of object descriptors that have been used since the server was last started.

4 Configure the number of open objects to 397, plus 10 percent (40), for a total of 437:

```
sp_configure "number of open objects", 437
```

If there is a lot of activity on the server, for example, if tables are being added or dropped, run sp_monitorconfig periodically. You will need to reset the cache size as the number of active descriptors changes. See sp_monitorconfig in the *Reference Manual* for more information.

open index hash spinlock ratio

Summary information	
Default value	100
Range of values	1–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

open index hash spinlock ratio sets the number of index metadata descriptor hash tables that are protected by one **spinlock**. This parameter is used for multiprocessing systems only.

All the index descriptors belonging to the table are accessible through a hash table. When a query is run on the table, Adaptive Server uses hash tables to look up the necessary index information in its sysindexes rows. A hash table is an internal mechanism used by Adaptive Server to retrieve information quickly.

Usually, you do not need to change this parameter. In rare instances, however, you may need to reset it if Adaptive Server demonstrates contention from hash spinlocks. You can get information about spinlock contention by using sp_sysmon. For more about sp_sysmon, see the *Performance and Tuning Guide*.

For more information about configuring spinlock ratios, see “Configuring spinlock ratio parameters” on page 668.

open index spinlock ratio

Summary information	
Default value	100
Range of values	1–214748364
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

open index spinlock ratio specifies the number of index metadata descriptors that are protected by one **spinlock**.

Adaptive Server uses a spinlock to protect an index descriptor, since more than one process can access the contents of the index descriptor. This parameter is used for multiprocessing systems only.

The value specified for this parameter defines the ratio of index descriptors per spinlock.

If one spinlock is shared by too many index descriptors, it can cause spinlock contention. Use sp_sysmon to get a report on spinlock contention. See the *Performance and Tuning Guide* more information. If sp_sysmon output indicates an index descriptor spinlock contention of more than 3 percent, try decreasing the value of open index spinlock ratio.

For more information about configuring spinlock ratios, see "Configuring spinlock ratio parameters" on page 668.

open object spinlock ratio

Summary information	
Default value	100
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

open object spinlock ratio specifies the number of object descriptors that are protected by one **spinlock**. Adaptive Server uses a spinlock to protect an object descriptor, since more than one process can access the contents of the object descriptor. This configuration parameter is used for multiprocessing systems only.

The default value for this parameter is 100; 1 spinlock for each 100 object descriptors configured for your server. If your server is configured with only one engine, Adaptive Server sets only 1 object descriptor spinlock, regardless of the number of object descriptors.

If one spinlock is shared by too many object descriptors, it causes spinlock contention. Use sp_sysmon to get a report on spinlock contention. See the *Performance and Tuning Guide* for more information on spinlock contention. If sp_sysmon output indicates an object descriptor spinlock contention of more than 3 percent, try decreasing the value of the open object spinlock ratio parameter.

For more information about configuring spinlock ratios, see "Configuring spinlock ratio parameters" on page 668.

Network communication

Use the parameters in this group to configure communication between Adaptive Server and remote servers, and between Adaptive Server and client programs.

allow remote access

Summary information	
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

allow remote access controls logins from remote Adaptive Servers. The default value of 1 allows Adaptive Server to communicate with Backup Server. Only a System Security Officer can set allow remote access.

Setting the value to 0 disables server-to-server RPCs. Since Adaptive Server communicates with Backup Server via RPCs, setting this parameter to 0 makes it impossible to back up a database.

Since other system administration actions are required to enable remote servers other than Backup Server to execute RPCs, leaving this option set to 1 does not constitute a security risk.

allow sendmsg

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer

The allow sendmsg parameter enables or disables sending messages from Adaptive Server to a UDP (User Datagram Protocol) port. When allow sendmsg is set to 1, any user can send messages using sp_sendmsg or syb_sendmsg. To set the port number used by Adaptive Server, see "syb_sendmsg port number" on page 131.

Note Sending messages to UDP ports is not supported on Windows NT.

default network packet size

Summary information	
Default value	512
Range of values	512–524288
Status	Static
Display level	Intermediate
Required role	System Administrator

default network packet size configures the default packet size for all Adaptive Server users. You can set default network packet size to any multiple of 512 bytes; values that are not even multiples of 512 are rounded down.

Memory for all users who log in with the default packet size is allocated from Adaptive Server's memory pool, as set with total logical memory. This memory is allocated for network packets when Adaptive Server is started.

Each Adaptive Server user connection uses:

- One read buffer
- One buffer for messages
- One write buffer

Each of these buffers requires default network packet size bytes. The total amount of memory allocated for network packets is:

```
(number of user connections + number of worker processes) * 3 * default network
packet size
```

For example, if you set the default network packet size to 1024 bytes, and you have 50 user connections and 20 worker processes, the amount of network memory required is:

(50 + 20) * 3 * 1024 = 215040 bytes

If you increase the default network packet size, you must also increase the max network packet size to at least the same size. If the value of max network packet size is greater than the value of default network packet size, to increase the value of additional network memory. See “additional network memory” on page 142 for further information.

Use sp_sysmon to see how changing the default network packet size parameter affects network I/O management and task switching. For example, try increasing default network packet size and then checking sp_sysmon output to see how this affects bcp for large batches. See the *Performance and Tuning Guide* for more information.

Requesting a larger packet size at login

The default packet size for most client programs like bcp and isql is set to 512 bytes. If you change the default packet size, clients must request the larger packet size when they connect. Use the -A flag to Adaptive Server client programs to request a large packet size. For example:

```
isql -A2048
```

max network packet size

Summary information	
Default value	512
Range of values	512–524288
Status	Static
Display level	Intermediate
Required role	System Administrator

max network packet size specifies the maximum network packet size that can be requested by clients communicating with Adaptive Server.

If some of your applications send or receive large amounts of data across the network, these applications can achieve significant performance improvement by using larger packet sizes. Two examples are large bulk copy operations and applications that read or write large text or image values.

Generally, you want:

- The value of default network packet size to be small for users who perform short queries
- max network packet size to be large enough to allow users who send or receive large volumes of data to request larger packet sizes

max network packet size must always be as large as, or larger than, the default network packet size. Values that are not even multiples of 512 are rounded down.

For client applications that explicitly request a larger network packet size to receive it, you must also configure additional network memory. See “additional network memory” on page 142 for more information.

Open Client Server cannot accept a network packet size greater than 64K.

See bcp and isql in the *Utility Guide* for information on using larger packet sizes from these programs. Open Client Client-Library documentation includes information on using variable packet sizes.

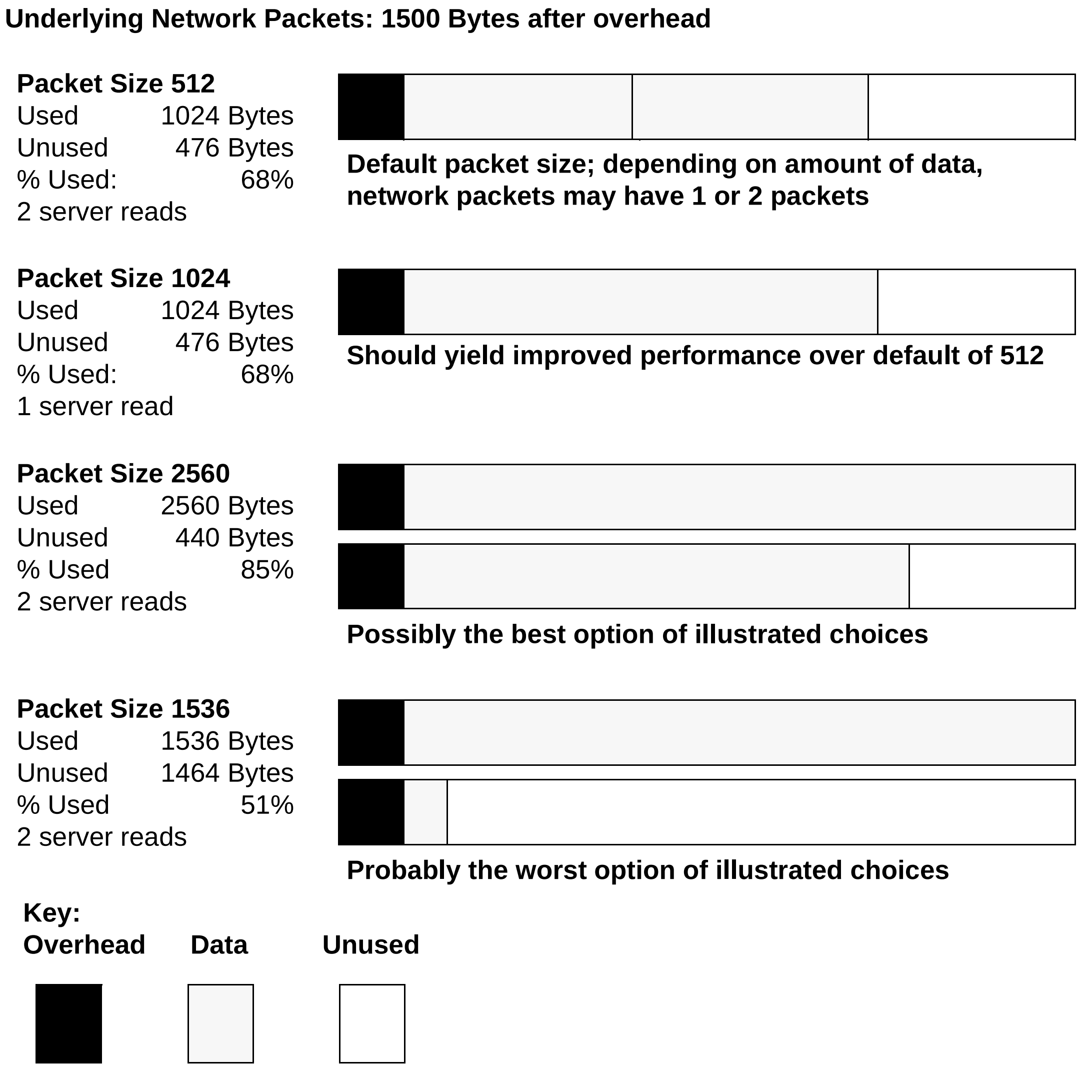
Choosing packet sizes

For best performance, choose a server packet size that works efficiently with the underlying packet size on your network. The goals are:

- Reducing the number of server reads and writes to the network
- Reducing unused space in network packets (increasing network throughput)

For example, if your network packet size carries 1500 bytes of data, setting Adaptive Server’s packet size to 1024 (512*2) will probably achieve better performance than setting it to 1536 (512*3). Figure 4-5 shows how four different packet size configurations would perform in such a scenario.

Figure 4-5: Factors in determining packet size



After you determine the available data space of the underlying packets on your network, perform your own benchmark tests to determine the optimum size for your configuration.

Use sp_sysmon to see how changing max network packet size affects network I/O management and task switching. For example, try increasing max network packet size and then checking sp_sysmon output to see how this affects bcp for large batches. See the *Performance and Tuning Guide* for more information.

max number network listeners

Summary information	
Default value	5
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

max number network listeners specifies the maximum number of network listeners allowed by Adaptive Server at one time.

Each master port has one network listener. Generally, there is no need to have multiple master ports, unless your Adaptive Server needs to communicate over more than one network type. Some platforms support both socket and TLI (Transport Layer Interface) network interfaces. Refer to the configuration documentation for your platform for information on supported network types.

number of remote connections

Summary information	
Default value	20
Range of values	5–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

number of remote connections specifies the number of logical connections that can be open to and from an Adaptive Server at one time. Each simultaneous connection to XP Server for ESP execution uses up to one remote connection each. For more information, see Chapter 13, “Managing Remote Servers.”

number of remote logins

Summary information	
Default value	20
Range of values	0–32767
Status	Static
Display level	Intermediate

Summary information	
Required role	System Administrator

number of remote logins controls the number of active user connections from Adaptive Server to remote servers. Each simultaneous connection to XP Server for ESP execution uses up to one remote login each. You should set this parameter to the same (or a lower) value as number of remote connections. For more information, see Chapter 13, “Managing Remote Servers.”

number of remote sites

Summary information	
Default value	10
Range of values	0–32767
Status	Static
Display level	Intermediate
Required role	System Administrator

number of remote sites determines the maximum number of remote sites that can access Adaptive Server simultaneously. An each Adaptive Server-to-XP Server connection uses one remote site connection.

Internally, number of remote sites determines the number of site handlers that can be active at any one time; all server accesses from a single site are managed with a single site handler. For more information, see Chapter 13, “Managing Remote Servers.”

remote server pre-read packets

Summary information	
Default value	3
Range of values	3–255
Status	Static
Display level	Intermediate
Required role	System Administrator

remote server pre-read packets determines the number of packets that will be “pre-read” by a site handler during connections with remote servers.

All communication between two servers is managed through a single site handler, to reduce the required number of connections. The site handler can pre-read and keep track of data packets for each user process before the receiving process is ready to accept them.

The default value for remote server pre-read packets is appropriate for most servers. Increasing the value uses more memory; decreasing the value can slow network traffic between servers. For more information, see Chapter 13, “Managing Remote Servers.”

syb_sendmsg port number

Summary information	
Default value	0
Valid values	0, or 1024–65535, or system limit
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The syb_sendmsg port number parameter specifies the port number that Adaptive Server uses to send messages to a UDP (User Datagram Protocol) port with sp_sendmsg or syb_sendmsg.

If more than one engine is configured, a port is used for each engine, numbered consecutively from the port number specified. If the port number is set to the default value, 0 Adaptive Server assigns port numbers.

Note Sending messages to UDP ports is not supported on Windows NT.

A System Security Officer must set the allow sendmsg configuration parameter to 1 to enable sending messages to UDP ports. To enable UDP messaging, a System Administrator must set allow sendmsg to 1. See “allow sendmsg” on page 125. For more information on UDP messaging, see sp_sendmsg in the *Reference Manual*.

tcp no delay

Summary information	
Default value	1 (on)
Valid values	0 (off), 1 (on)

Summary information	
Status	Static
Display level	Comprehensive
Required role	System Administrator

The tcp no delay parameter controls TCP (Transmission Control Protocol) packet batching. The default value is 1, which means that TCP packets are not batched.

TCP normally batches small logical packets into single larger physical packets (by briefly delaying packets) fill physical network frames with as much data as possible. This is intended to improve network throughput in terminal emulation environments where there are mostly keystrokes being sent across the network.

However, applications that use small TDS (Tabular Data Stream™) packets may benefit from disabling TCP packet batching. To disable TCP packet batching, set tcp no delay to 1.

Note Disabling TCP packet batching means that packets will be sent, regardless of size; this will increase the volume of network traffic.

O/S resources

The parameters in this group are related to Adaptive Server's use of operating system resources.

max async i/os per engine

Summary information	
Default value	2147483647
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

max async i/os per engine specifies the maximum number of outstanding asynchronous disk I/O requests for a single engine at one time. See "max async i/os per server" on page 133 for more information.

max async i/os per server

Summary information	
Default value	2147483647
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The max async i/os per server parameter specifies the maximum number of asynchronous disk I/O requests that can be outstanding for Adaptive Server at one time. This limit is not affected by the number of online engines per Adaptive Server; max async i/os per server limits the total number of asynchronous I/Os a server can issue at one time, regardless of how many online engines it has. max async i/os per engine limits the number of outstanding I/Os per engine.

Most operating systems limit the number of asynchronous disk I/Os that can be processed at any one time; some operating systems limit the number per operating system process, some limit the number per system, and some do both. If an application exceeds these limits, the operating system returns an error message. Because operating system calls are relatively expensive, it is inefficient for Adaptive Server to attempt to perform asynchronous I/Os that get rejected by the operating system.

To avoid this, Adaptive Server maintains a count of the outstanding asynchronous I/Os per engine and per server; if an engine issues an asynchronous I/O that would exceed either max async i/os per engine or max async i/os per server, Adaptive Server delays the I/O until enough outstanding I/Os have completed to fall below the exceeded limit.

For example, assume an operating system limit of 200 asynchronous I/Os per system and 75 per process and an Adaptive Server with three online engines. The engines currently have a total of 200 asynchronous I/Os pending, distributed according to the following table:

Engine	Number of I/Os pending	Outcome
0	60	Engine 0 delays any further asynchronous I/Os until the total for the server is under the operating system *per-system* limit and then continues issuing asynchronous I/Os.
1	75	Engine 1 delays any further asynchronous I/Os until the per-engine total is under the operating system *per-process* limit and then continues issuing asynchronous I/Os.
2	65	Engine 2 delays any further asynchronous I/Os until the total for server is under the operating system *per-system* limit and then continues issuing asynchronous I/Os.

All I/Os (both asynchronous and synchronous) require a disk I/O structure, so the total number of outstanding disk I/Os is limited by the value of disk i/o structures. It is slightly more efficient for Adaptive Server to delay the I/O because it cannot get a disk I/O structure than because the I/O request exceeds max i/os per server. You should set max async i/os per server equal to the value of disk i/o structures. See “disk i/o structures” on page 86.

If the limits for asynchronous I/O can be tuned on your operating system, make sure they are set high enough for Adaptive Server. There is no penalty for setting them as high as needed.

Use sp_sysmon to see if the per server or per engine limits are delaying I/O on your system. If sp_sysmon shows that Adaptive Server exceeded the limit for outstanding requests per engine or per server, raise the value of the corresponding parameter. See the *Performance and Tuning Guide* for more information.

o/s file descriptors

Summary information	
Default value	0
Range of values	Site-specific
Status	Read-only
Display level	Comprehensive
Required role	System Administrator

o/s file descriptors indicates the maximum per-process number of file descriptors configured for your operating system. This parameter is read-only and cannot be configured through Adaptive Server.

Many operating systems allow you to configure the number of file descriptors available per process. See your operating system documentation for further information on this.

The number of file descriptors available for Adaptive Server connections, which will be less than the value of o/s file descriptors, is stored in the variable *@@max_connections*. For more information on the number of file descriptors available for connections, see “Upper limit to the maximum number of user connections” on page 197.

shared memory starting address

Summary information	
Default value	0
Range of values	Platform-specific
Status	Static
Display level	Comprehensive
Required role	System Administrator

shared memory starting address determines the virtual address where Adaptive Server starts its shared memory region.

It is unlikely that you will ever have to reconfigure shared memory starting address. You should do so only after consulting with Sybase Technical Support.

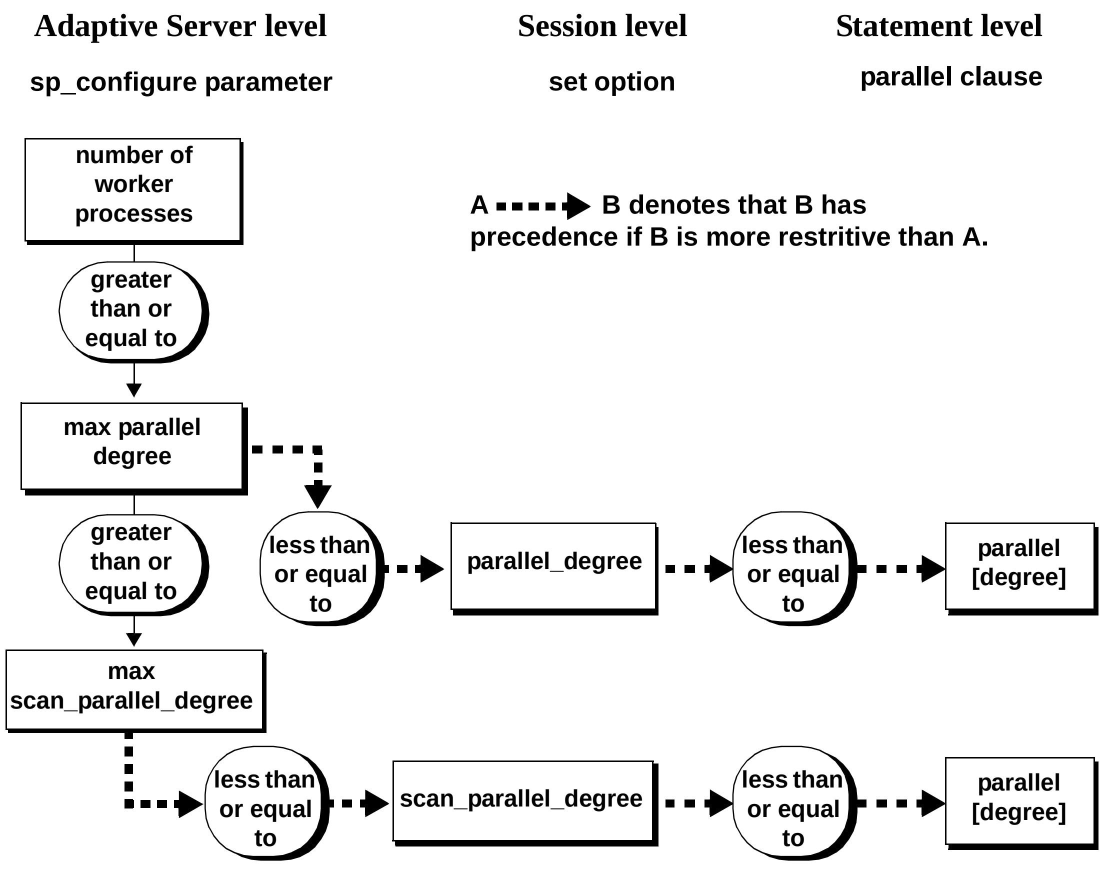
Parallel queries

The following parameters configure Adaptive Server for parallel query processing – where the optimizer evaluates each query to determine whether it is eligible for parallel execution.

To determine the best values to use for the configuration parameters, and to understand how these values affect the optimizer, see Chapter 2, "Optimizer Overview" and Chapter 8, "Parallel Query Optimization" in the *Performance and Tuning Guide: Optimizer and Abstract Plans*.

number of worker processes, max parallel degree, and max scan parallel degree control parallel query processing at the server level. Using the parallel_degree, process_limit_action, and scan_parallel_degree options to the set command can limit parallel optimization at the session level, and using the parallel keyword of the select command can limit parallel optimization of specific queries. Figure 4-6 shows the precedence of the configuration parameters and session parameters.

Figure 4-6: Precedence of parallel options



number of worker processes

Summary information	
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

number of worker processes specifies the maximum number of worker processes that Adaptive Server can use at any one time for all simultaneously running parallel queries combined.

Adaptive Server issues a warning message at start-up if there is insufficient memory to create the specified number of worker processes. memory per worker process controls the memory allocated to each worker process.

max parallel degree

Summary information	
Default value	1
Range of values	1–255
Status	Dynamic
Display level	Basic
Required role	System Administrator

max parallel degree specifies the server-wide maximum number of worker processes allowed per query. This is called the *maximum degree of parallelism*.

If this number is too low, the performance gain for a given query may not be as significant as it could be; if the number is too high, the server may compile plans that require more processes than are actually available at execution time, or the system may become saturated, resulting in decreased throughput. To enable parallel partition scans, set this parameter to be equal to or greater than the number of partitions in the table you are querying.

The value of this parameter must be less than or equal to the current value of number of worker processes.

If you set max parallel degree to 1, Adaptive Server scans all tables or indexes serially.

Changing max parallel degree causes all query plans in the procedure cache to be invalidated, and new plans are compiled the next time you execute a stored procedure or trigger.

For more information on parallel sorting, see Chapter 9, “Parallel Sorting” in the *Performance and Tuning Guide: Optimizer and Abstract Plans.*

max scan parallel degree

Summary information	
Default value	1
Range of values	1–255
Status	Dynamic
Display level	Basic
Required role	System Administrator

max scan parallel degree specifies the server-wide maximum degree of parallelism for hash-based scans. Hash-based scans may be used for the following access methods:

- Parallel index scans for partitioned and nonpartitioned tables
- Parallel table scans for nonpartitioned tables

max scan parallel degree applies per table or index; that is, if max scan parallel degree is 3, and one table in a join query is scanned using a hash-based table scan and the second can best be accessed by a hash-based index scan, the query could use 9 worker processes (as long as max scan parallel degree is set to 9 or higher.)

The optimizer uses this parameter as a guideline when it selects the number of processes to use for parallel, nonpartition-based scan operations. It does not apply to parallel sort. Because there is no partitioning to spread the data across devices, parallel processes can be accessing the same device during the scan. This can cause additional disk contention and head movement, which can degrade performance. To prevent multiple disk accesses from becoming a problem, use this parameter to reduce the maximum number of processes that can access the table in parallel.

If this number is too low, the performance gain for a given query will not be as significant as it could be; if the number is too large, the server may compile plans that use enough processes to make disk access less efficient. A general rule of thumb is to set this parameter to no more than 2 or 3, because it takes only 2 to 3 worker processes to fully utilize the I/O of a given physical device.

Set the value of this parameter to less than or equal to the current value of max parallel degree. Adaptive Server returns an error if you specify a number larger than the max parallel degree value.

If you set max scan parallel degree to 1, Adaptive Server does not perform hash-based scans.

Changing max scan parallel degree causes all query plans in the procedure cache to be invalidated, and new plans are compiled the next time you execute a stored procedure or trigger.

memory per worker process

Summary information	
Default value	1024
Range of values	1024–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

memory per worker process specifies the amount of memory (in bytes) used by worker processes. Each worker process requires memory for messaging during query processing. This memory is allocated from a shared memory pool; the size of this pool is memory per worker process multiplied by number of worker processes. For most query processing, the default size is more than adequate. If you use dbcc checkstorage, and have set number of worker processes to 1, you may need to increase memory per worker process to 1792 bytes. See Chapter 9, “Parallel Sorting” of the *Performance and Tuning Guide: Optimizer and Abstract Plans* for information on setting this parameter.

For more information on Adaptive Server’s memory allocation, see Chapter 18, “Configuring Memory.”

Physical memory

The parameters in this group configure your machine’s physical memory resources.

allocate max shared memory

Summary information	
Default value	0
Range of values	0,1
Status	Dynamic
Display level	Basic
Required role	System Administrator

allocate max shared memory determines whether Adaptive Server allocates all the memory specified by max memory at start-up or only the amount of memory the configuration parameter requires.

By setting allocate max shared memory to 0, you ensure that Adaptive Server uses only the amount of shared memory required by the current configuration, and allocates only the amount of memory required by the configuration parameters at start-up, which is a smaller value than max memory.

If you set allocate max shared memory to 1, Adaptive Server allocates all the memory specified by max memory at start-up. If allocate max shared memory is 1, and if you increase max memory, Adaptive Server immediately uses additional shared memory segments. This means that Adaptive Server always has the memory required for any memory configuration changes you make and there is no performance degradation while the server readjusts for additional memory. However, if you do not predict memory growth accurately, and max memory is set to a large value, you may waste total physical memory.

dynamic allocation on demand

Summary information	
Default value	1
Range of values	0, 1
Status	Dynamic
Display level	Basic
Required role	System Administrator

Determines when memory is allocated for changes to dynamic memory configuration parameters.

If you set dynamic allocation on demand to 1, memory is allocated only as it is needed. That is, if you change the configuration for number of user connections from 100 to 200, the memory for each user is added only when the user connects to the server. Adaptive Server continues to add memory until it reaches the new maximum for user connections.

If dynamic allocation on demand is set to 0, all the memory required for any dynamic configuration changes is allocated immediately. That is, when you change the number of user connections from 100 to 200, the memory required for the extra 100 user connections is immediately allocated.

max memory

Summary information	
Default value	Platform-dependent
Range of values	Platform-dependent minimum – 2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

Specifies the maximum amount of total physical memory that you can configure Adaptive Server to allocate. max memory must be greater than the total logical memory consumed by the current configuration of Adaptive Server.

There is no performance penalty for configuring Adaptive Server to use the maximum memory available to it on your computer. However, assess the other memory needs on your system, or Adaptive Server may not be able to acquire enough memory to start.

See Chapter 18, “Configuring Memory,” in the *System Administration Guide* for instructions on how to maximize the amount of max memory for Adaptive Server.

If Adaptive Server cannot start

When allocate max shared memory is set to 1, Adaptive Server must have the amount of memory available that is specified by max memory. If the memory is not available, Adaptive Server will not start. If this occurs, reduce the memory requirements for Adaptive Server by manually changing the value of max memory in the server’s configuration file. You can also change the value of allocate max shared memory to 0 so that not all memory required by max memory is required at start-up.

You may also want to reduce the values for other configuration parameters that require large amounts of memory. Then restart Adaptive Server to use the memory specified by the new values. If Adaptive Server fails to start because the total of other configuration parameter values is higher than the max memory value, see Chapter 18, “Configuring Memory,” in the *System Administration Guide* for information about configuration parameters that use memory.

additional network memory

Summary information	
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

additional network memory sets the maximum size of additional memory that can be used for network packets that are larger than the default packet size. Adaptive Server rounds down the value you enter to the nearest 2K value. The default value indicates that no extra space is allocated for large packets.

If you increase max network packet size but do not increase additional network memory, clients cannot use packet sizes that are larger than the default size, because all allocated network memory is reserved for users at the default size. Adaptive Server guarantees that every user connection can log in at the default packet size. In this situation, users who request a large packet size when they log in receive a warning message telling them that their application will use the default size.

Increasing additional network memory may improve performance for applications that transfer large amounts of data. To determine the value for additional network memory when your applications use larger packet sizes:

- Estimate the number of simultaneous users who will request the large packet sizes, and the sizes their applications will request,
- Multiply this sum by three, since each connection needs three buffers,
- Add two percent for overhead, and
- Round the value to the next highest multiple of 2048.

For example, if you estimate these simultaneous needs for larger packet sizes:

Application	Packet size	Overhead
bcp	8192	
Client-Library	8192	
Client-Library	4096	
Client-Library	4096	
Total	24576	
Multiply by 3 buffers/user	* 3=73728	
Compute 2% overhead		* .02=1474

Application	Packet size	Overhead
Add overhead	+ 1474	
Additional network memory	75202	
Round up to multiple of 2048	75776	

You should set additional network memory to 75,776 bytes.

heap memory per user

Summary information	
Default value	4K
Valid values	0 – 2147483647 bytes
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

heap memory per user configures the amount of heap memory per user. A heap memory pool is an internal memory created at start-up that tasks use to dynamically allocate memory as needed. This memory pool is important if you are running tasks that use wide columns, which require a lot of memory from the stack. The heap memory allocates a temporary buffer that enables these wide column tasks to finish. The heap memory the task uses is returned to the heap memory pool when the task is finished.

The size of the memory pool depends on the number of user connections. Sybase recommends that you set heap memory per user to three times the size of your logical page.

lock shared memory

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

lock shared memory disallows swapping of Adaptive Server pages to disk and allows the operating system kernel to avoid the server's internal page locking code. This can reduce disk reads, which are expensive.

Not all platforms support shared memory locking. Even if your platform does, lock shared memory may fail due to incorrectly set permissions, insufficient physical memory, or for other reasons. See the configuration documentation for your platform for information on shared memory locking.

max SQL text monitored

Summary information	
Default value	0
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

max SQL text monitored specifies the amount of memory allocated per user connection for saving SQL text to memory shared by Adaptive Server Monitor.

Initially, the amount of memory allocated for saving text is 0, and since this parameter is static, you must restart Adaptive Server before you can start saving SQL Text.

If you do not allocate enough memory for the batch statements, the text you want to view may be in the section of the batch that is truncated. Sybase recommends an initial value of 1024 bytes of memory per user connection.

The total memory allocated from shared memory for the SQL text is the product of max SQL text monitored multiplied by the currently configured number of user connections.

For more information on max SQL text monitored, see “Configuring Adaptive Server to save SQL batch text” on page 227.

total physical memory

Summary information	
Default value	N/A
Range of values	N/A
Status	Read-only
Display level	Intermediate
Required role	System Administrator

total physical memory is a read-only configuration parameter that displays the total physical memory for the current configuration of Adaptive Server. The total physical memory is the amount of memory that Adaptive Server is using at a given moment in time. Adaptive Server should be configured so that the value for max memory is larger than the value for total logical memory, and the value for total logical memory is larger than the value for total physical memory.

total logical memory

Summary information	
Default value	N/A
Range of values	N/A
Status	Read-only
Display level	Intermediate
Required role	System Administrator

total logical memory displays the total logical memory for the current configuration of Adaptive Server. The total logical memory is the amount of memory that Adaptive Server's current configuration uses. total logical memory displays the memory which is required to be available, but which may or may not be in use at any given moment. For information about the amount of memory in use at a given moment, see the configuration parameter total physical memory. You cannot use total logical memory to set any of the memory configuration parameters.

Processors

The parameters in this group configure processors in an SMP environment.

number of engines at startup

Summary information	
Default value	1
Range of values	1 – number of CPUs on machine
Status	Static
Display level	Basic
Required role	System Administrator

Adaptive Server allows users to take all engines offline, except engine zero.

number of engines at startup is used exclusively during start-up to set the number of engines brought online. It is designed to allow users the greatest flexibility in the number of engines brought online, subject to the restriction that you cannot set the value of number of engines at startup to a value greater than the number of CPUs on your machine, or to a value greater than the configuration of max online engines. Users who do not intend to bring engines online after start-up should set max online engines and number of engines at startup to the same value. A difference between number of engines at startup and max online engines wastes approximately 1.8 MB of memory per engine.

max online engines

Summary information	
Default value	1
Range of values	1–128
Status	Static
Display level	Intermediate
Required role	System Administrator

The role of max online engines is to set a high value of engines to be taken online at any one time in an SMP environment. It does not take the number of CPUs available at start-up into account, and allows users to add CPUs at a later date.

max engines online specifies the maximum number of Adaptive Server engines that can be online at any one time in an SMP environment. See Chapter 20, “Managing Multiprocessor Servers,” for a detailed discussion of how to set this parameter for your SMP environment.

At start-up, Adaptive Server starts with a single engine and completes its initialization, including recovery of all databases. Its final task is to allocate additional server engines. Each engine accesses common data structures in shared memory.

When tuning the max engines online parameter:

- Never have more online engines than there are CPUs.
- Depending on overall system load (including applications other than Adaptive Server), you may achieve optimal throughput by leaving some CPUs free to run non-Adaptive Server processes.

- Better throughput can be achieved by running fewer engines with high CPU use, rather than by running more engines with low CPU use.
- Scalability is application-dependent. You should conduct extensive benchmarks on your application to determine the best configuration of online engines.
- You can use the dbcc engine command to take engines offline or to bring them online line. You can offline all engines other than engine zero.

See “Taking engines offline with dbcc engine” on page 664 for information on using dbcc engine. See Chapter 4, “Using Engines and CPUs” in the *Performance and Tuning: Basics* for more information on performance and engine tuning.

RepAgent thread administration

The parameter in this group configures replication via Replication Server®.

enable rep agent threads

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Basic
Required role	System Administrator

enable rep agent threads enables the RepAgent thread within Adaptive Server.

Through version 11.0.3 of Replication Server, the Log Transfer Manager (LTM), a replication system component, transfers replication data to the Replication Server. Beginning with Replication Server versions later than 11.0.3, transfer of replication data handled by RepAgent, which will run as a thread under Adaptive Server. Setting enable rep agent threads enables this feature.

Other steps are also required to enable replication. For more information, see the Replication Server documentation.

SQL server administration

The parameters in this group are related to general Adaptive Server administration.

abstract plan cache

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

abstract plan cache enables caching of abstract plan hash keys. By default, caching is not enabled. For more information, see Chapter 16, “Creating and Using Abstract Plans” in *Performance and Tuning: Optimizer and Abstract Plans*. abstract plan load must be enabled in order for plan caching to take affect.

abstract plan dump

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

abstract plan dump enables the saving of abstract plans to the ap_stdout abstract plans group. For more information, see Chapter 16, “Creating and Using Abstract Plans” in *Performance and Tuning: Optimizer and Abstract Plans*.

abstract plan load

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic

Summary information	
Display level	Comprehensive
Required role	System Administrator

abstract plan load enables association of queries with abstract plans in the ap_stdin abstract plans group. For more information, see Chapter 16, "Creating and Using Abstract Plans" in *Performance and Tuning: Optimizer and Abstract Plans*.

abstract plan replace

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

abstract plan replace enables plan replacement for abstract plans in the ap_stdout abstract plans group. For more information, see Chapter 16, "Creating and Using Abstract Plans" in *Performance and Tuning: Optimizer and Abstract Plans*. abstract plan load must be enabled in order for replace mode to take effect.

allow backward scans

Summary information	
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

allow backward scans controls how the optimizer performs select queries that contain the order by...desc command:

- When the value is set to 1, the optimizer can access the index or table rows by following the page chain in descending index order.

- When the value is set to 0, the optimizer selects the rows into a worktable by following the index page pointers in ascending order and then sorts the worktable in descending order.

The first method—performing backward scans—can speed access to tables that need results ordered by descending column values. Some applications, however, may experience deadlocks due to backward scans. In particular, look for increased deadlocking if you have delete or update queries that scan forward using the same index. There may also be deadlocks due to page splits in the index.

You can use print deadlock information to send messages about deadlocks to the error log. See "print deadlock information" on page 174. Alternatively, you can use sp_sysmon to check for deadlocking. See the *Performance and Tuning Guide* for more information on deadlocks.

allow nested triggers

Summary information	
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Intermediate
Required role	System Administrator

allow nested triggers controls the use of nested triggers. When the value is set to 1, data modifications made by triggers can fire other triggers. Set allow nested triggers to 0 to disable nested triggers. A set option, self_recursion, controls whether the modifications made by a trigger can cause that trigger to fire again.

allow resource limits

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

allow resource limits controls the use of resource limits. When the value is set to 1, the server allocates internal memory for time ranges, resource limits, and internal server alarms. The server also internally assigns applicable ranges and limits to user sessions. The output of showplan and statistics io displays the optimizer's cost estimate for a query. Set allow resource limits to 0 to disable resource limits.

allow updates to system tables

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

allow updates to system tables enables users with the System Administrator role to make changes to the system tables and to create stored procedures that can modify system tables. A database administrator can update system tables in any tables that he or she owns if allow updates to system tables is enabled.

System tables include:

- All Sybase-supplied tables in the master database
- All tables in user databases that begin with "sys" and that have an ID value in the sysobjects table of less than or equal to 100

Warning! Incorrect alteration of a system table can result in database corruption and loss of data. Always use begin transaction when modifying a system table to protect against errors that could corrupt your databases. Immediately after finishing your modifications, disable allow updates to system tables.

Stored procedures and triggers you create while allow updates to system tables is set on are always able to update the system tables, even after the parameter has been set off. When you set allow updates to system tables to on, you create a "window of vulnerability," a period of time during which users can alter system tables or create a stored procedure with which the system tables can be altered in the future.

Because the system tables are so critical, it is best to set this parameter to on only in highly controlled situations. To guarantee that no other users can access Adaptive Server while the system tables can be directly updated, restart Adaptive Server in single-user mode. For details, see startserver and dataserver in the *Utility Guide*.

cpu accounting flush interval

Summary information	
Default value	200
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

cpu accounting flush interval specifies the amount of time, in *machine* clock ticks, that Adaptive Server waits before flushing CPU usage statistics for each user from sysprocesses to syslogins, a procedure used in chargeback accounting. (Note that this is measured in *machine* clock ticks, not Adaptive Server clock ticks.)

When a user logs in to Adaptive Server, the server begins accumulating figures for CPU usage for that user process in sysprocesses. When a user logs off Adaptive Server, or when the value of cpu accounting flush interval is exceeded, the accumulated CPU usage statistics are flushed from sysprocesses to syslogins. These statistics continue accumulating in syslogins until you clear the totals using sp_clearstats. You can display the current totals from syslogins using sp_reportstats.

The value to which you set cpu accounting flush interval depends on the type of reporting you intend to do. If you intend to run reports on a monthly basis, set cpu accounting flush interval to a relatively high value. With infrequent reporting, it is less critical that the data in syslogins be updated as often.

On the other hand, if you intend to do periodic ad hoc selects on the totcpu column in syslogins to determine CPU usage by process, set cpu accounting flush interval to a lower value. Doing so increases the likelihood of the data in syslogins being up to date when you execute your selects.

Setting cpu accounting flush interval to a low value may cause processes to be mistakenly identified as potential deadlock victims by the lock manager. When the lock manager detects a deadlock, it checks the amount of CPU time accumulated by each competing processes. The process with the lesser amount is chosen as the deadlock victim and is terminated by the lock manager. Additionally, when cpu accounting flush interval is set to a low value, the task handlers that store CPU usage information for processes are initialized more frequently, thus making processes appear as if they have accumulated less CPU time than they actually have. Because of this, the lock manager may select a process as the deadlock victim when, in fact, that process has more accumulated CPU time than the competing process.

If you do not intend to report on CPU usage at all, set cpu accounting flush interval to its maximum value. This reduces the number of times syslogins is updated and reduces the number of times its pages need to be written to disk.

cpu grace time

Summary information	
Default value	500
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

cpu grace time, together with time slice, specifies the maximum amount of time that a user process can run without yielding the CPU before Adaptive Server preempts it and terminates it with a time-slice error. The units for cpu grace time are time ticks, as defined by sql server clock tick length. See “sql server clock tick length” on page 177 for more information.

When a process exceeds cpu grace time Adaptive Server “infects” it by removing the process from the internal queues. The process is killed, but Adaptive Server is not affected. This prevents runaway processes from monopolizing the CPU. If any of your user processes become infected, you may be able to temporarily fix the problem by increasing the value of cpu grace time. However, you must be sure that the problem really is a process that takes more than the current value of cpu grace time to complete, rather than a runaway process.

Temporarily increasing the cpu grace time value is a workaround, not a permanent fix, since it may cause other complications; see "time slice" on page 179. Also, See Chapter 4, "Using Engines and CPUs" in the *Performance and Tuning: Basics* for a more detailed discussion of task scheduling.

default database size

Summary information	
Default value	Logical page size
Range of values	2[a] –10000 a. Minimum determined by server's logical page size.
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

default database size sets the default number of megabytes allocated to a new user database if the create database statement is issued without any size parameters. A database size given in a create database statement takes precedence over the value set by this configuration parameter.

If most of the new databases on your Adaptive Server require more than one logical page size, you may want to increase the default.

Note If you alter the model database, you must also increase the default database size, because the create database command copies model to create a new user database.

default fill factor percent

Summary information	
Default value	0
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

default fill factor percent determines how full Adaptive Server makes each index page when it is creating a new index on existing data, unless the fill factor is specified in the create index statement. The fillfactor percentage is relevant only at the time the index is created. As the data changes, the pages are not maintained at any particular level of fullness.

default fill factor percent affects:

- The amount of storage space used by your data – Adaptive Server redistributes the data as it creates the clustered index.
- Performance – splitting up pages uses Adaptive Server resources.

There is seldom a reason to change default fill factor percent, especially since you can override it in the create index command. For more information about the fill factor percentage, see “create index” in the *Reference Manual*.

default exp_row_size percent

Summary information	
Default value	5
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

default exp_row_size percent reserves space for expanding updates in data-only-locked tables, to reduce row forwarding. An *expanding update* is any update to a data row that increases the length of the row. Data rows that allow null values or that have variable-length columns may be subject to expanding updates. In data-only-locked tables, expanding updates can require row forwarding if the data row increases in size so that it no longer fits on the page.

The default value, sets aside 5 percent of the available data page size for use by expanding updates. Since 2002 bytes are available for data storage on pages in data-only-locked tables, this leaves 100 bytes for expansion. This value is only applied to pages for tables that have variable-length columns.

Valid values are 0–99. Setting default exp_row_size percent to 0 means that all pages are completely filled and no space is left for expanding updates.

default exp_row_size percent is applied to data-only-locked tables with variable-length columns when exp_row_size is not explicitly provided with create table or set with sp_chgattribute. If a value is provided with create table, that value takes precedence over the configuration parameter setting. See the *Performance and Tuning Guide* for more information.

dump on conditions

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

dump on conditions determines whether Adaptive Server generates a dump of data in shared memory when it encounters the conditions specified in maximum dump conditions.

Note The dump on conditions parameter is included for use by Sybase Technical Support only. Do not modify it unless you are instructed to do so by Sybase Technical Support.

enable sort-merge joins and JTC

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

enable sort-merge joins and JTC configuration parameter determines whether merge joins and join transitive closure are considered by the query optimizer. By default, merge joins and join transitive closure are not enabled. To enable merge joins, set this parameter to 1.

Merge joins and join transitive closure can improve performance for queries that access large amounts of data, but increase optimization time. The session-level options set sort-merge on and set jtc on take precedence over the server-wide setting. For more information, see Chapter 3, "Advanced Optimizing Tools" in *Performance and Tuning: Optimizer and Abstract Plans*.

enable row level access control

Summary information	
Default value	0
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer

Enables row level access control. You must have the security services license key enabled before you configure enable row level access control.

enable ssl

Summary information	
Default value	0
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Security Officer

The enable ssl parameter enables or disables Secure Sockets Layer session-based security.

event buffers per engine

Summary information	
Default value	100
Range of values	1–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

The event buffers per engine parameter specifies the number of events per Adaptive Server engine that can be monitored simultaneously by Adaptive Server Monitor. Events are used by Adaptive Server Monitor for observing Adaptive Server performance; if you are not using Adaptive Server Monitor, set this parameter to 1.

The value to which you set event buffers per engine depends on the number of engines in your configuration, the level of activity on your Adaptive Server, and the kinds of applications you are running.

Setting event buffers per engine to a low value may result in the loss of event information. The default value, is likely to be too low for most sites. Values of 2000 and above may be more reasonable for general monitoring. However, you need to experiment to determine the appropriate value for your site.

In general, setting event buffers per engine to a high value may reduce the amount of performance degradation that Adaptive Server Monitor causes Adaptive Server.

Each event buffer uses 100 bytes of memory. To determine the total amount of memory used by a particular value for event buffers per engine, multiply the value by the number of Adaptive Server engines in your configuration.

housekeeper free write percent

Summary information	
Default value	1
Range of values	0–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

housekeeper free write percent specifies the maximum percentage by which the housekeeper wash task can increase database writes.

For example, to stop the housekeeper task from working when the frequency of database writes reaches 5 percent above normal, set housekeeper free write percent to 5:

```
sp_configure "housekeeper free write percent", 5
```

When Adaptive Server has no user tasks to process, the housekeeper wash task automatically begins writing changed pages from cache to disk. These writes result in improved CPU utilization, decreased need for buffer washing during transaction processing, and shorter checkpoints.

In applications that repeatedly update the same database page, the housekeeper wash may initiate some unnecessary database writes. Although these writes occur only during the server's idle cycles, they may be unacceptable on systems with overloaded disks.

The table and index statistics that are used to optimize queries are maintained in memory structures during query processing. When these statistics change, the changes are not written to the systabstats table immediately, to reduce I/O contention and improve performance. Instead, the housekeeper chores task periodically flushes statistics to disk.

The default value allows the housekeeper wash task to increase disk I/O by a maximum of 1 percent. This results in improved performance and recovery speed on most systems.

To disable the housekeeper wash task, set the value of housekeeper free write percent to 0:

```
sp_configure "housekeeper free write percent", 0
```

You should set this value to 0 only if disk contention on your system is high, and it cannot tolerate the extra I/O generated by the housekeeper wash task.

If you disable the housekeeper tasks, be certain that statistics are kept current. Commands that write statistics to disk are:

- update statistics
- dbcc checkdb (for all tables in a database) or dbcc checktable (for a single table)
- sp_flushstats

You should run one of these commands on any tables that have been updated since the last time statistics were written to disk, at the following times:

- Before dumping a database
- Before an orderly shutdown
- After rebooting, following a failure or orderly shutdown; in these cases, you cannot use sp_flushstats, you must use update statistics or dbcc commands
- After any significant changes to a table, such as a large bulk copy operation, altering the locking scheme, deleting or inserting large numbers of rows, or a truncate table command

To allow the housekeeper wash task to work continuously, regardless of the percentage of additional database writes, set housekeeper free write percent to 100:

```
sp_configure "housekeeper free write percent", 100
```

Use sp_sysmon to monitor housekeeper performance. See the *Performance and Tuning Guide* for more information.

It might also be helpful to look at the number of free checkpoints initiated by the housekeeper task. The *Performance and Tuning Guide* describes this output.

enable HA

Summary information	
Default value	0
Range of values	0–1
Status	Static
Display level	Comprehensive
Required role	System Administrator

Setting enable HA is set to 1 allows you to configure Adaptive Server as a companion server in a high availability subsystem. Adaptive Server use s Sybase's Failover to interact with the high availability subsystem. You must set enable HA to 1 before you run the *installhasvss* script (*insthasv* on Windows NT), which installs the system procedures for Sybase's Failover.

Note The license information and the Run value for enable HA are independent of each other. Whether or not you have a license for Sybase Failover, the Run value and the Config value are set to 1 after you reboot Adaptive Server. And until you have a license, you cannot run Sybase Failover. If you have not installed a valid license, Adaptive Server logs an error message and does not activate the feature. See your *Installation Guide* for information about installing license keys.

Note that, setting enable HA to 1 does not mean that Adaptive Server is configured to work in a high availability system. You must perform the steps described in *Using Sybase Failover in A High Availability System* to configure Adaptive Server to be a companion server in a high availability system.

When enable HA is set to 0, you cannot configure for Sybase's Failover, and you cannot run *installhasvss* (*insthasv* on Windows NT).

enable housekeeper GC

Summary information	
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The housekeeper garbage collection task performs space reclamation on data-only-locked tables. When a user task deletes a row from a data-only-locked table, a task is queued to the housekeeper to check the data and index pages for committed deletes.

The housekeeper garbage collection task is controlled by enable housekeeper GC. For more information on the housekeeper garbage collection task see Chapter 4, "Using Engines and CPUs" in the *Performance and Tuning: Basics*.

sp_sysmon reports on how often the housekeeper garbage collection task performed space reclamation and how many pages were reclaimed. See the *Performance and Tuning Guide* for more information.

identity burning set factor

Summary information	
Default value	5000
Range of values	1–9999999
Status	Static
Display level	Intermediate
Required role	System Administrator

IDENTITY columns are of type numeric and scale zero whose values are generated by Adaptive Server. Column values can range from a low of 1 to a high determined by the column precision.

For each table with an IDENTITY column, Adaptive Server divides the set of possible column values into blocks of consecutive numbers, and makes one block at a time available in memory. Each time you insert a row into a table, Adaptive Server assigns the IDENTITY column the next available value from the block. When all the numbers in a block have been used, the next block becomes available.

This method of choosing IDENTITY column values improves server performance. When Adaptive Server assigns a new column value, it reads the current maximum value from memory and adds 1. Disk access becomes necessary only after all values within the block have been used. Because all remaining numbers in a block are discarded in the event of server failure (or shutdown with nowait), this method can lead to gaps in IDENTITY column values.

Use identity burning set factor to change the percentage of potential column values that is made available in each block. This number should be high enough for good performance, but not so high that gaps in column values are unacceptably large. The default value, 5000, releases .05 percent of the potential IDENTITY column values for use at one time.

To get the correct value for sp_configure, express the percentage in decimal form, and then multiply it by 10^7 (10,000,000). For example, to release 15 percent (.15) of the potential IDENTITY column values at a time, specify a value of .15 times 10^7 (or 1,500,000) in sp_configure:

```
sp_configure "identity burning set factor", 1500000
```

identity grab size

Summary information	
Default value	1
Range of values	1–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

identity grab size allows each Adaptive Server process to reserve a block of IDENTITY column values for inserts into tables that have an IDENTITY column.

This is useful if you are doing inserts, and you want all the inserted data to have contiguous IDENTITY numbers. For instance, if you are entering payroll data, and you want all records associated with a particular department to be located within the same block of rows, set identity grab size to the number of records for that department.

identity grab size applies to all users on Adaptive Server. Large identity grab size values result in large gaps in the IDENTITY column when many users insert data into tables with IDENTITY columns.

Sybase recommends setting identity grab size to a value large enough to accommodate the largest group of records you want to insert into contiguous rows.

i/o accounting flush interval

Summary information	
Default value	1000
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

i/o accounting flush interval specifies the amount of time, in *machine* clock ticks, that Adaptive Server waits before flushing I/O statistics for each user from sysprocesses to syslogins. This is used for chargeback accounting.

When a user logs in to Adaptive Server, the server begins accumulating I/O statistics for that user process in sysprocesses. When the value of i/o accounting statistics interval is exceeded, or a user logs off Adaptive Server, the accumulated I/O statistics for that user are flushed from sysprocesses to syslogins. These statistics continue accumulating in syslogins until you clear the totals by using sp_clearstats. You display the current totals from syslogins by using sp_reportstats.

The value to which you set i/o accounting flush interval depends on the type of reporting you intend to do. If you intend to run reports on a monthly basis, i/o accounting flush interval to a relatively high value. This is because, with infrequent reporting, it is less critical that the data in syslogins be updated frequently.

If you intend to do periodic ad hoc selects on the totio column syslogins to determine I/O volume by process, to set i/o accounting flush interval to a lower value. Doing so increases the likelihood of the data in syslogins being up to date when you execute your selects.

If you do not intend to report on I/O statistics at all, set i/o accounting flush interval to its maximum value. This reduces the number of times syslogins is updated and the number of times its pages need to be written to disk.

i/o polling process count

Summary information	
Default value	10
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

i/o polling process count specifies the maximum number of processes that can be run by Adaptive Server before the scheduler checks for disk and/or network I/O completions. Tuning i/o polling process count affects both the response time and throughput of Adaptive Server.

Adaptive Server checks for disk or network I/O completions:

- If the number of tasks run since the last time Adaptive Server checked for I/O completions equals the value for i/o polling process count, and
- At every Adaptive Server clock tick.

As a general rule, increasing the value of i/o polling process count may increase throughput for applications that generate a lot of disk and network I/O. Conversely, decreasing the value may improve process response time in these applications, possibly at the risk of lowering throughput.

If your applications create both I/O and CPU-bound tasks, tuning i/o polling process count to a low value (1–2) ensures that I/O-bound tasks get access to CPU cycles.

For OLTP applications (or any I/O-bound application with user connections and short transactions), tuning i/o polling process count to a value in the range of 20–30 may increase throughput, but it may also increase response time.

When tuning i/o polling process count, consider three other parameters:

- sql server clock tick length, which specifies the duration of Adaptive Server's clock tick in microseconds. See "sql server clock tick length" on page 177.
- time slice, which specifies the number of clock ticks Adaptive Server's scheduler allows a user process to run. See "time slice" on page 179.
- cpu grace time, which specifies the maximum amount of time (in clock ticks) a user process can run without yielding the CPU before Adaptive Server preempts it and terminates it with a time-slice error. See "cpu grace time" on page 153.

Use sp_sysmon to determine the effect of changing the i/o polling process count parameter. See the *Performance and Tuning Guide* for more information.

page lock promotion HWM

Summary information	
Default value	200
Range of values	2–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

page lock promotion HWM (high-water mark), together with the page lock promotion LWM (low-water mark) and page lock promotion PCT (percentage), specifies the number of page locks permitted during a single scan session of a page-locked table or index before Adaptive Server attempts to escalate from page locks to a table lock.

page lock promotion HWM sets a maximum number of page locks allowed on a table before Adaptive Server attempts to escalate to a table lock. When the number of page locks acquired during a scan session exceeds page lock promotion HWM, Adaptive Server attempts to acquire a table lock. The page lock promotion HWM value cannot be higher than number of locks value.

For more detailed information on scan sessions and setting up page lock promotion limits, see "Configuring locks and lock promotion thresholds" in the *Performance and Tuning: Locking*.

The default value for page lock promotion HWM is appropriate for most applications. You might want to raise the value to avoid table locking. For example, if you know that there are regular updates to 500 pages of an allpages-locked or datapages-locked table containing thousands of pages, you can increase concurrency for the tables by setting page lock promotion HWM to 500 so that lock promotion does not occur at the default setting of 200.

You can also configure lock promotion of page-locked tables and views at the per-object level. See sp_setrowlockpromote in the *Reference Manual*.

Use sp_sysmon to see how changing page lock promotion HWM affects the number of lock promotions. sp_sysmon reports the ratio of exclusive page to exclusive table lock promotions and the ratio of shared page to shared table lock promotions. See "Lock promotions" in the *Performance and Tuning: Monitoring and Analyzing*.

page lock promotion LWM

Summary information	
Default value	200
Range of values	2–value of page lock promotion HWM
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

The page lock promotion LWM low-water mark) parameter, together with the page lock promotion HWM (high-water mark) and the page lock promotion PCT, specify the number of page locks permitted during a single scan session of a page locked table or an index before Adaptive Server attempts to promote from page locks to a table lock.

The page lock promotion LWM sets the number of page locks below which Adaptive Server does not attempt to issue a table lock on an object. The page lock promotion LWM must be less than or equal to page lock promotion HWM.

For more information on scan sessions and setting up lock promotion limits, see "Configuring locks and lock promotion thresholds" in the *Performance and Tuning: Locking*.

The default value for page lock promotion LWM is sufficient for most applications. If Adaptive Server runs out of locks (except for an isolated incident), you should increase number of locks. See the *Performance and Tuning Guide* for more information.

You can also configure page lock promotion at the per-object level. See sp_setpglockpromote in the *Reference Manual: Procedures*.

page lock promotion PCT

Summary information	
Default value	100
Range of values	1–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

If the number of locks held on an object is between page lock promotion LWM (low-water mark) and page lock promotion HWM (high-water mark). page lock promotion PCT sets the percentage of page locks (based on the table size) above which Adaptive Server attempts to acquire a table lock.

For more detailed information on setting up page lock promotion limits, see “Configuring locks and lock promotion thresholds” in *Performance and Tuning: Locking*.

The default value for page lock promotion PCT is appropriate for most applications.

You can also configure lock promotion at the per-object level for page locked objects. See sp_setpglockpromote in the *Reference Manual*.

maximum dump conditions

Summary information	
Default value	10
Range of values	10–100
Status	Static
Display level	Intermediate
Required role	System Administrator

The maximum dump conditions parameter sets the maximum number of conditions you can specify under which Adaptive Server generates a dump of data in shared memory.

Note This parameter is included for use by Sybase Technical Support only. Do not modify it unless you are instructed to do so by Sybase Technical Support.

---

number of alarms

Summary information	
Default value	40
Range of values	40–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

number of alarms specifies the number of alarm structures allocated by Adaptive Server.

The Transact-SQL command waitfor defines a specific time, time interval, or event for the execution of a statement block, stored procedure, or transaction. Adaptive Server uses alarms to execute waitfor commands correctly. Other internal processes require alarms.

When Adaptive Server needs more alarms than are currently allocated, this message is written to the error log:

```
uasetalarm: no more alarms available
```

The number of bytes of memory required for each is small. If you raise the number of alarms value significantly, you should adjust max memory accordingly.

number of aux scan descriptors

Summary information	
Default value	200
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive

Summary information	
Required role	System Administrator

number of aux scan descriptors sets the number of auxiliary scan descriptors available in a pool shared by all users on a server.

Each user connection and each worker process has 48 scan descriptors exclusively allocated to it. Of these, 16 are reserved for user tables, 12 are reserved for worktables, and 20 are reserved for system tables (with 4 of these set aside for rollback conditions). A descriptor is needed for each table referenced, directly or indirectly, by a query. For user tables, a table reference includes the following:

- All tables referenced in the from clause of the query
- All tables referenced in a view named in the query (the view itself is not counted)
- All tables referenced in a subquery
- All tables that need to be checked for referential integrity (these are used only for inserts, updates, and deletes)
- A table created with select...into
- All worktables created for the query

If a table is referenced more than once (for example, in a self-join, in more than one view, or in more than one subquery) the table is counted each time. If the query includes a union, each select statement in the union query is a separate scan. If a query runs in parallel, the coordinating process and each worker process needs a scan descriptor for each table reference.

When the number of user tables referenced by a query scan exceeds 16, or the number of worktables exceeds 12, scan descriptors from the shared pool are allocated. Data-only-locked tables also require a system table descriptor for each data-only-locked table accessed via a table scan (but not those accessed via an index scan). If more than 16 data-only-locked tables are scanned using table scans in a query, auxiliary scan descriptors are allocated for them.

If a scan needs auxiliary scan descriptors after it has used its allotted number, and there are no descriptors available in the shared pool, Adaptive Server displays an error message and rolls back the user transaction.

If none of your queries need additional scan descriptors, you may still want to leave number of aux scan descriptors set to the default value in case your system requirements grow. Set it to 0 only if you are sure that users on your system will not be running queries on more than 16 tables and that your tables have few or no referential integrity constraints. See "Monitoring scan descriptor usage" on page 170 for more information.

If your queries need more scan descriptors, use one of the following methods to remedy the problem:

- Rewrite the query, or break it into steps using temporary tables. For data-only-locked tables, consider adding indexes if there are many table scans.
- Redesign the table's schema so that it uses fewer scan descriptors, if it uses a large number of referential integrity constraints. You can find how many scan descriptors a query would use by enabling set showplan, noexec on before running the query.
- Increase the number of aux scan descriptors setting.

The following sections describe how to monitor the current and high-water-mark usage with sp_monitorconfig to avoid running out of descriptors and how to estimate the number of scan descriptors you need.

Monitoring scan descriptor usage

sp_monitorconfig reports the number of unused (free) scan descriptors, the number of auxiliary scan descriptors currently being used, the percentage that is active, and the maximum number of scan descriptors used since the server was last started. Run it periodically, at peak periods, to monitor scan descriptor use.

This example output shows scan descriptor use with 500 descriptors configured:

```
sp_monitorconfig "aux scan descriptors"
```

```
Usage information at date and time: Apr 22 2002  2:49PM.
Name                num_free  num_active pct_act         Max_Used Reused
--------------      --------  ---------  --------        -------- ------
number of aux            260        240  48.00                427  NA
```

Only 240 auxiliary scan descriptors are being used, leaving 260 free. However, the maximum number of scan descriptors used at any one time since the last time Adaptive Server was started is 427, leaving about 20 percent for growth in use and exceptionally heavy use periods. "Re-used" does not apply to scan descriptors.

Estimating and configuring auxiliary scan descriptors

To get an estimate of scan descriptor use:

1. Determine the number of table references for any query referencing more than 16 user tables or those that have a large number of referential constraints, by running the query with set showplan and set noexec enabled. If auxiliary scan descriptors are required, showplan reports the number needed:

   ```
   Auxiliary scan descriptors required: 17
   ```

   The reported number includes all auxiliary scan descriptors required for the query, including those for all worker processes. If your queries involve only referential constraints, you can also use sp_helpconstraint, which displays a count of the number of referential constraints per table.

2. For each query that uses auxiliary scan descriptors, estimate the number of users who would run the query simultaneously and multiply. If 10 users are expected to run a query that requires 8 auxiliary descriptors, a total of 80 will be needed at any one time.

3. Add the per-query results to calculate the number of needed auxiliary scan descriptors.

number of mailboxes

Summary information	
Default value	30
Range of values	30–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

number of mailboxes specifies the number of mailbox structures allocated by Adaptive Server. Mailboxes, which are used in conjunction with messages, are used internally by Adaptive Server for communication and synchronization between kernel service processes. Mailboxes are not used by user processes. Do not modify this parameter unless instructed to do so by Sybase Technical Support.

number of messages

Summary information	
Default value	64
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

number of messages specifies the number of message structures allocated by Adaptive Server. Messages, which are used in conjunction with mailboxes, are used internally by Adaptive Server for communication and synchronization between kernel service processes. Messages are also used for coordination between a family of processes in parallel processing. Do not modify this parameter unless instructed to do so by Sybase Technical Support.

number of pre-allocated extents

Summary information	
Default value	2
Range of values	1–31
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

number of pre-allocated extents specifies the number of extents (eight pages) allocated in a single trip to the page manager. Currently, it is used only by bcp to improve performance when copying in large amounts of data. By default, bcp allocates two extents at a time and writes an allocation record to the log each time.

Setting number of pre-allocated extents means that bcp allocates the specified number of extents each time it requires more space, and writes a single log record for the event.

An object may be allocated more pages than actually needed, so the value of number of pre-allocated extents should be low if you are using bcp for small batches. If you are using bcp for large batches, increase the value of number of pre-allocated extents to reduce the amount of overhead required to allocate pages and to reduce the number of log records.

number of sort buffers

Summary information	
Default value	500
Range of values	0–32767
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

number of sort buffers specifies the number of 2K buffers used to hold pages read from input tables and perform index merges during sorts.

Sybase recommends that you leave this parameter set to the default except when you are creating indexes in parallel. Setting the value too high can rob non-sorting processes of access to the 2K buffer pool in caches being used to perform sorts.

For more information on configuring this value for parallel create index statements, see “Caches, sort buffers, and parallel sorts” in the *Performance and Tuning: Optimizer and Abstract Plans*.

partition groups

Summary information	
Default value	1024
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

partition groups specifies the maximum number of partition groups that can be allocated by Adaptive Server. Partition groups are internal structures used by Adaptive Server to control access to individual partitions of a table.

A partition group is composed of 16 partition caches, each of which stores information about a single partition. All caches in a partition group are used to store information about the same partitioned table. If a table has fewer than 16 partitions, the unused partition caches in that group are unused, and cannot be used by another table. If a table has more than 16 partitions, it requires multiple partition groups.

The default value allows a maximum 1024 open partition groups and a maximum of 16384 (1024 times 16) open partitions. The actual number of partitions may be slightly less, due to the grouping of partitions.

Adaptive Server allocates partition groups to a table when you partition the table or when you access it for the first time after restarting Adaptive Server. If there are not enough partition groups for the table, you will not be able to access or partition the table.

partition spinlock ratio

Summary information	
Default value	10
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

For Adaptive Servers running with multiple engines, partition spinlock ratio sets the number of rows in the internal partition caches that are protected by one spinlock.

Adaptive Server manages access to table partitions using internal *partition groups*, each of which contains partition caches. Each partition cache stores information about a partition (for example, the last page of the partition) that processes must use when accessing that partition.

By default, Adaptive Server systems are configured with partition spinlock ratio set to 10, or 1 spinlock for every 10 partition caches. Decreasing the value of partition spinlock ratio may have little impact on the performance of Adaptive Server. The default setting is correct for most servers.

For more information about configuring spinlock ratios, see “Configuring spinlock ratio parameters” on page 668.

print deadlock information

Summary information	
Default value	0 (off)
Valid values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate

Summary information	
Required role	System Administrator

print deadlock information enables the printing of deadlock information to the error log.

If you are experiencing recurring deadlocks, setting print deadlock information to 1 provides you with information that can be useful in tracing the cause of the deadlocks. However, setting print deadlock information to 1 can seriously degrade Adaptive Server performance. For this reason, you should use it only when you are trying to determine the cause of deadlocks.

Use sp_sysmon output to determine whether deadlocks are occurring in your application. If they are, set print deadlock information to 1 to learn more about why they are occurring. See the *Performance and Tuning Guide* for more information.

runnable process search count

Summary information	
Default value	2000
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

runnable process search count specifies the number of times an engine loops while looking for a runnable task before relinquishing the CPU to the operating system.

Adaptive Server engines check the run queue for runnable tasks whenever a task completes or exceeds its allotted time on the engine. At times, there will not be any tasks in the run queues. An engine can either relinquish the CPU to the operating system or continue to check for a task to run. Setting runnable process search count higher causes the engine to loop more times, thus holding the CPU for a longer time. Setting the runnable process search count lower causes the engine to release the CPU sooner.

If your machine is a uniprocessor that depends on helper threads to perform I/O, you may see some performance benefit from setting runnable process search order to perform network I/O, disk I/O, or other operating system tasks. If a client, such as a bulk copy operation, is running on the same machine as a single CPU server that uses helper threads, it can be especially important to allow both the server and the client access to the CPU.

Note If you are having performance problems, try setting runnable process search count to 3.

For Adaptive Servers running on uniprocessor machines that do not use helper threads, and for multiprocessor machines, the default value provides good performance.

Use sp_sysmon to determine how the runnable process search count parameter affects Adaptive Server's use of CPU cycles, engine yields to the operating system, and blocking network checks. See the *Performance and Tuning Guide* for information.

size of auto identity column

Summary information	
Default value	10
Range of values	1–38
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

size of auto identity column sets the precision of IDENTITY columns that are automatically created with the sp_dboption auto identity and unique auto_identity index options.

The maximum value that can be inserted into an IDENTITY column is $10^{precision}$ -1. After an IDENTITY column reaches its maximum value, all further insert statements return an error that aborts the current transaction.

If you reach the maximum value of an IDENTITY column, you can increase it with a modify operation in the alter table command. See *Transact-SQL User's Guide* for examples.

You can also use the create table command to create a table that is identical to the old one, but with a larger precision for the IDENTITY column. After you have created the new table, use the insert command or bcp to copy data from the old table to the new one.

SQL Perfmon Integration (Windows NT only)

Summary information	
Default value	1 (on)
Valid values	0 (off), 1 (on)
Status	Static
Display level	Intermediate
Required role	System Administrator

SQL Perfmon Integration enables and disables the ability to monitor Adaptive Server statistics from the Windows NT Performance Monitor.

Adaptive Server must be registered as an NT Service to support monitor integration. This occurs automatically when:

- You start Adaptive Server using the Services Manager in the Sybase for Windows NT program group.
- You use the Services option in the Control Panel.
- You have configured Windows NT to start Adaptive Server as an automatic service.

See *Configuring Adaptive Server for Windows NT* for a list of the Adaptive Server counters you can monitor.

sql server clock tick length

Summary information	
Default value	Platform-specific
Range of values	Platform-specific minimum–1000000, in multiples of default value
Status	Static
Display level	Comprehensive
Required role	System Administrator

sql server clock tick length specifies the duration of the server's clock tick, in microseconds. Both the default value and the minimum value are platform-specific. Adaptive Server rounds values up to an even multiple of *n*, where *n* is the platform-specific clock-tick default value. You can find the current values for sql server clock tick length by using sp_helpconfig or sp_configure.

In mixed-use applications with some CPU-bound tasks, decreasing the value of sql server clock tick length helps I/O-bound tasks. A value of 20,000 is reasonable for this. Shortening the clock tick length means that CPU-bound tasks will exceed the allotted time on the engine more frequently per unit of time, which allows other tasks greater access to the CPU. This may also marginally increase response times, because Adaptive Server runs its service tasks once per clock tick. Decreasing the clock tick length means that the service tasks will be run more frequently per unit of time.

Increasing sql server clock tick length favors CPU-bound tasks, because they execute longer between context switches. The maximum value of 1,000,000 may be appropriate for primarily CPU-bound applications. However, any I/O-bound tasks may suffer as a result. This can be mitigated somewhat by tuning cpu grace time (see "cpu grace time" on page 153) and time slice (see "time slice" on page 179).

Note Changing the value of sql server clock tick length can have serious effects on Adaptive Server's performance. You should consult with Sybase Technical Support before resetting this value.

text prefetch size

Summary information	
Default value	16
Valid values	0–65535
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The text prefetch size parameter limits the number of pages of text and image data that can be prefetched into an existing buffer pool. Adaptive Server prefetches only text and image data that was created with Adaptive Server 12.x or was upgraded using dbcc rebuild_text.

time slice

Summary information	
Default value	100
Range of values	50–1000
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

time slice sets the number of milliseconds that Adaptive Server’s scheduler allows a task to run. If time slice is set too low, Adaptive Server may spend too much time switching between tasks, which increases response time. If it is set too high, CPU-intensive tasks might monopolize engines, which also increases response time. The default value, 100 milliseconds, allows each task to run for 1/10 of a second before relinquishing the CPU to another task.

See “cpu grace time” and Chapter 4, “Using Engines and CPUs” in the *Performance and Tuning: Basics* for a more detailed discussion of task scheduling.

Use sp_sysmon to determine how time slice affects voluntary yields by Adaptive Server engines. See the *Performance and Tuning Guide* for more information.

upgrade version

Summary information	
Default value	1100
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

upgrade version reports the version of the upgrade utility that upgraded your master device. The upgrade utility checks and modifies this parameter during an upgrade.

Warning! Although this parameter is configurable, you should not reset it. Doing so may cause serious problems with Adaptive Server.

You can determine whether an upgrade has been done on your master device by using upgrade version without specifying a value:

```
sp_configure "upgrade version"
```

row lock promotion HWM

Summary information	
Default value	200
Range of values	2–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

row lock promotion HWM (high-water mark), together with row lock promotion LWM (low-water mark) and row lock promotion PCT specifies the number of row locks permitted during a single scan session of a table or an index before Adaptive Server attempts to escalate from row locks to a table lock.

row lock promotion HWM sets a maximum number of row locks allowed on a table before Adaptive Server attempts to escalate to a table lock. When the number of locks acquired during a scan session exceeds row lock promotion HWM, Adaptive Server attempts to acquire a table lock. The lock promotion HWM value cannot be higher than the number of locks value.

For more information on scan sessions and setting up lock promotion limits, see “Configuring locks and lock promotion thresholds” in *Performance and Tuning: Locking*.

The default value for row lock promotion HWM is appropriate for most applications. You might want to raise the value to avoid table locking. For example, if you know that there are regular updates to 500 rows on a table that has thousands of rows, you can increase concurrency for the tables by setting row lock promotion HWM to around 500.

You can also configure row lock promotion at the per-object level. See sp_setpglockpromote in the *Reference Manual*.

row lock promotion LWM

Summary information	
Default value	200

Summary information	
Range of values	2–value of row lock promotion HWM
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

row lock promotion LWM (low-water mark), together with the row lock promotion HWM (high-water mark) and row lock promotion PCT specifies the number of row locks permitted during a single scan session of a table or an index before Adaptive Server attempts to promote from row locks to a table lock.

row lock promotion LWM sets the number of locks below which Adaptive Server does not attempt to acquire a table lock on the object. The row lock promotion LWM must be less than or equal to row lock promotion HWM.

For more detailed information on scan sessions and setting up lock promotion limits, see “Configuring locks and lock promotion thresholds” in *Performance and Tuning: Locking*.

The default value for row lock promotion LWM is sufficient for most applications. If Adaptive Server runs out of locks (except for an isolated incident), you should increase number of locks. See the *Performance and Tuning Guide* for more information.

You can also configure lock promotion at the per-object level. See sp_setpglockpromote in the *Reference Manual*.

row lock promotion PCT

Summary information	
Default value	100
Range of values	1–100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

If the number of locks held on an object is between row lock promotion LWM (low-water mark) and row lock promotion HWM (high-water mark), row lock promotion PCT sets the percentage of row locks (based on the number of rows in the table) above which Adaptive Server attempts to acquire a table lock.

For more information on setting up lock promotion limits, see “Configuring locks and lock promotion thresholds” in *Performance and Tuning: Locking*.

The default value for row lock promotion PCT is appropriate for most applications.

You can also configure row lock promotion at the per-object level. See sp_sterowlockpromote in the *Reference Manual*.

license information

Summary information	
Default value	25
Valid values	0–2^{31}
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

license information allows Sybase System Administrators to monitor the number of user licenses used in Adaptive Server. Enabling this parameter only monitors the number of licenses issued; it does not enforce the license agreement.

If license information is set to 0, Adaptive Server does not monitor license use. If license information is set to a number greater than 0, the housekeeper chores task monitors the number of licenses used during the idle cycles in Adaptive Server. Set license information to the number of licenses specified in your license agreement.

license information is set to 25, by default. To disable license information, issue the command:

```
sp_configure "license information", 0
```

If the number of licenses used is greater than the number to which license information is set, Adaptive Server writes the following error message to the error log:

```
WARNING: Exceeded configured number of user licenses
```

At the end of each 24-hour period, the maximum number of licenses used during that time is added to the syblicenseslog table. The 24-hour period restarts if Adaptive Server is restarted.

See “Monitoring license use” on page 381 for more information.

Security related

The parameters in this group configure security-related features.

allow procedure grouping

Summary information	
Default value	1 (on)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer

allow procedure grouping controls the ability to group stored procedures of the same name so that they can be dropped with a single drop procedure statement. To run Adaptive Server in the *evaluated configuration*, you must prohibit stored procedure grouping by setting this option to 0. See evaluated configuration in the *Adaptive Server Glossary* for more information.

auditing

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

auditing enables or disables auditing for Adaptive Server.

maximum failed logins

This section provides information about maximum failed logins, a new configuration parameter.

Summary information

Name in pre-12.0 version	N/A
Default value	0
Range of values	0 – 32767
Status	Dynamic

Summary information

Display level	10
Required role	System Security Officer

maximum failed logins allows you to set the server-wide maximum number of failed login attempts for logins and roles. For example, to set the system-wide maximum failed logins to 5, enter:

```
sp_configure "maximum failed logins", 5
```

Use create role to set maximum failed logins for a specific role or creation. To create the intern_role role with the password “temp244”, and set maximum failed logins for intern_role to 20, enter:

```
create role intern_role with passwd "temp244", maximum
failed logins 20
```

Use sp_modifylogin to set or change maximum failed logins for an existing login. To change maximum failed logins for the login “joe” to 40, enter:

```
sp_modifylogin "joe", @option="maximum failed logins",
@value="40"
```

Note The *value* parameter is a character datatype; therefore, quotes are required for numeric values.

To change the overrides for maximum failed logins for all logins to 3, enter:

```
sp_modifylogin "all overrides", "maximum failed
logins", "3"
```

To remove the overrides for maximum failed logins option for all logins, enter:

```
sp_modifylogin "all overrides", @option="maximum failed
logins", @value="-1"
```

Use alter role to set or change the maximum failed logins for an existing role. For example, to change the maximum failed logins allowed for physician_role to 5, enter:

```
alter role physician_role set maximum failed logins 5
```

To remove the overrides for maximum failed logins for all roles, enter:

```
alter role "all overrides" set maximum failed logins -1
```

check password for digit

This section provides information about check password for digit, a new configuration parameter.

Summary information

Name in pre-12.0 version	N/A
Default value	0
Range of values	1, 0
Status	Dynamic
Display level	10
Required role	System Security Officer

The System Security Officer can tell the server to check for at least one character or 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.

To activate check password for digit functionality, enter:

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

To deactivate check password for digit functionality, enter:

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

minimum password length

This section provides information about minimum password length, a new configuration parameter.

Summary information

Name in pre-12.0 version	minimum password length
Default value	6
Range of values	0 – 30
Status	Dynamic
Display level	10
Required role	System Security Officer

minimum password length allows you to customize the length of server-wide password values or per-login or per-role password values to fit your personal needs. The per-login or per-role minimum password length value overrides the server-wide value. Setting minimum password length affects only the passwords you create after you have set the value; existing password lengths are not changed.

Use minimum password length to specify a server-wide value for minimum password length for both logins and roles. For example, to set the minimum password length for all logins and roles to 4 characters, enter:

```
sp_configure "minimum password length", 4
```

To set minimum password length for a specific login at creation, use sp_addlogin. For example, to create the new login “joe” with the password “Djdiek3”, and set minimum password length for “joe” to 4, enter:

```
sp_addlogin joe, "Djdiek3", minimum password length=4
```

To set minimum password length for a specific role at creation, use create role. To create the new role “intern_role” with the password “temp244” and set the minimum password length for “intern_role” to 0, enter:

```
create role intern_role with passwd "temp244", minimum
password length 0
```

The original password is seven characters, but the password can be changed to one of any length because the minimum password length is set to 0.

Use sp_modifylogin to set or change minimum password length for an existing login. sp_modifylogin only effects user roles, not system roles. For example, to change minimum password length for the login “joe” to 8 characters, enter:

```
sp_modifylogin "joe", @option="minimum password
length", @value="8"
```

Note The *value* parameter is a character datatype; therefore, quotes are required for numeric values.

To change the value of the overrides for minimum password length for all logins to 2 characters, enter:

```
sp_modifylogin "all overrides", "minimum password
length", @value="2"
```

To remove the overrides for minimum password length for all logins, enter:

```
sp_modifylogin "all overrides", @option="minimum
password length", @value="-1"
```

Use alter role to set or change the minimum password length for an existing role. For example, to set the minimum password length for “physician_role”, an existing role, to 5 characters, enter:

```
alter role physician_role set minimum password length 5
```

To override the minimum password length for all roles, enter:

```
alter role "all overrides" set minimum password length
-1
```

audit queue size

Summary information	
Default value	100
Range of values	1–65535
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The in-memory audit queue holds audit records generated by user processes until the records can be processed and written to the audit trail. A System Security Officer can change the size of the audit queue with audit queue size. There is a trade-off between performance and risk that must be considered when you configure the queue size. If the queue is too large, records can remain in it for some time. As long as records are in the queue, they are at risk of being lost if the system crashes. However, if the queue is too small, it can become full repeatedly, which affects overall system performance–user processes that generate audit records sleep if the audit queue is full.

Following are some guidelines for determining how big your audit queue should be. You must also take into account the amount of auditing to be done at your site.

- The memory requirement for a single audit record is 424 bytes; however a record can be as small as 22 bytes when it is written to a data page
- The maximum number of audit records that can be lost in a system crash is the size of the audit queue (in records), plus 20. After records leave the audit queue they remain on a buffer page until they are written to the current audit table on the disk. The pages are flushed to disk every 20 records at the most (less if the audit process is not constantly busy).

- In the system audit tables, the extrainfo field and fields containing names are of variable length, so audit records that contain complete name information are generally larger.

The number of audit records that can fit on a page varies from 4 to as many as 80 or more. The memory requirement for the default audit queue size of 100 is approximately 42K.

current audit table

Summary information	
Default value	1
Range of values	0–8
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

current audit table establishes the table where Adaptive Server writes audit rows. A System Security Officer can change the current audit table, using:

```
sp_configure "current audit table", n
  [, "with truncate"]
```

where n is an integer that determines the new current audit table, as follows:

- 1 means sysaudits_01, 2 means sysaudits_02, and so forth, up to 8.
- 0 tells Adaptive Server to set the current audit table to the next table. For example, if your installation has three audit tables, sysaudits_01, sysaudits_02, and sysaudits_03, Adaptive Server sets the current audit table to:
  - 2 if the current audit table is sysaudits_01
  - 3 if the current audit table is sysaudits_02
  - 1 if the current audit table is sysaudits_03

"with truncate" specifies that Adaptive Server should truncate the new table if it is not already empty. sp_configure fails if this option is not specified and the table is not empty.

Note If Adaptive Server truncates the current audit table, and you have not archived the data, the table's audit records are lost. Be sure that the audit data is archived before using the with truncate option.

To execute sp_configure to change the current audit table, you must have the sso_role active. You can write a threshold procedure to change the current audit table automatically.

enable ssl

Summary information	
Default value	0
Valid values	0 (off), 1 (on)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The enable ssl parameter enables or disables Secure Sockets Layer session-based security.

msg confidentiality reqd

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

The msg confidentiality reqd parameter requires that all messages into and out of Adaptive Server be encrypted. The use security services parameter must be 1 for messages to be encrypted.

msg integrity reqd

Summary information	
Default value	0 (off)
Range of values	0 (off), 1 (on)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

msg integrity reqd requires that all messages be checked for data integrity. use security services must be 1 for message integrity checks to occur. If msg integrity reqd is set to one, Adaptive Server allows the client connection to succeed unless the client is using one of the following security services: message integrity, replay detection, origin checks, or out-of-seq checks.

secure default login

Summary information	
Default value	0
Range of values	0 (followed by another parameter naming the default login)
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

secure default login specifies a default login for all users who are preauthenticated but who do not have a login in master..syslogins.

Establish the secure default login with:

```
sp_configure "secure default login", 0,
    default_login_name
```

where:

- secure default login – is the name of the parameter.
- 0 – is a required parameter because the second parameter of sp_configure must be a numeric value.
- *default_login_name* – is the name of the default login for a user who is unknown to Adaptive Server, but who has already been authenticated by a security mechanism. The login name must be a valid login in master..syslogins.

For example, to specify “dlogin” as the secure default login, type:

```
sp_configure "secure default login", 0, dlogin
```

select on syscomments.text column

Summary information	
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer

This parameter enables protection of the text of database objects through restriction of the select permission on the text column of the syscomments table. The default value of 1 allows select permission to “public.” Set the option to 0 to restrict select permission to the object owner and the System Administrator.

To run Adaptive Server in the *evaluated configuration*, you must protect the source text of database objects by setting this option to 0.

See evaluated configuration in the *Adaptive Server Glossary* for more information.

suspend audit when device full

Summary information	
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

suspend audit when device full determines what Adaptive Server does when an audit device becomes completely full.

Note If you have two or more audit tables, each on a separate device other than the master device, and you 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.

Choose one of these values:

- 0 – truncates the next audit table and starts using it as the current audit table when the current audit table becomes full. If you set the parameter to 0, you ensure that the audit process is never suspended. However, you incur the risk that older audit records will get lost if they have not been archived.
- 1 – 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.

To run in the evaluated configuration, set this parameter to 1. See evaluated configuration in the *Adaptive Server Glossary* for more information.

systemwide password expiration

Summary information	
Default value	0
Range of values	0–32767
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

systemwide password expiration, which can be set only by a System Security Officer, sets the number of days that passwords remain in effect after they are changed. If systemwide password expiration is set to 0, passwords do not expire. If it is set to a number greater than 0, all passwords expire after the specified number of days. An account's password is considered expired if an interval greater than *number_of_days* has passed since the last time the password for that account was changed.

When the number of days remaining before expiration is less than 25 percent of the value of systemwide password expiration or 7 days, whichever is greater, each time the user logs in, a message displays, giving the number of days remaining before expiration. Users can change their passwords anytime before expiration.

When an account's password has expired, the user can still log in to Adaptive Server but cannot execute any commands until he or she has used sp_password to change his or her password. If the System Security Officer changes the user's password while the account is in sp_password-only mode, the account returns to normal after the new password is assigned.

This restriction applies only to login sessions established after the password has expired. Users who are logged in at the time their passwords expire are not affected until the next time they log in.

unified login required (Windows NT only)

Summary information	
Default value	0
Range of values	0, 1
Status	Dynamic
Display level	Intermediate
Required role	System Security Officer

unified login required requires that all users who log in to Adaptive Server be authenticated by the Windows NT LAN Manager. The use security services parameter must be 1 to use the unified login security service.

use security services (Windows NT only)

Summary information	
Default value	0

Summary information	
Range of values	0, 1
Status	Static
Display level	Intermediate
Required role	System Security Officer

use security services specifies that Adaptive Server will use security services provided by Windows NT LAN Manager. If the parameter is set to 0, unified login services with the LAN Manager cannot be used.

enable ldap user auth

Summary information	
Default value	0 (off)
Valid values	0 (off)–allows only syslogins authentication, 1 (on)–allows both LDAP and syslogins authentication, 2 (on)–allows only LDAP authentication
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer

When enable ldap user auth is 1, Adaptive Server searches the LDAP server to authenticate each user. If the LDAP authentication fails, Adaptive Server searches syslogins to authenticate the user. Use this level when migrating users from Adaptive Server authentication to LDAP authentication.

Unicode

The parameters in this group configure Unicode-related features.

default unicode sortorder

Summary information	
Default value	0
Range of values	(not currently used)
Status	Static
Display level	Comprehensive
Required role	System Administrator

The default unicode sortorder parameter is a string parameter that defines the default Unicode sort order installed on the server. A string parameter is used rather than a numeric parameter to guarantee a unique ID. To change the Unicode default sort order, see Chapter 7, “Configuring Character Sets, Sort Orders, and Languages.”

enable surrogate processing

Summary information	
Default value	1
Range of values	0 – 1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

Activates the processing and maintains the integrity of surrogate pairs in Unicode data. Set enable surrogate processing to 1 to enable surrogate processing. If this is disabled, the server ignores the presence of surrogate pairs in the Unicode data, and all code that maintains the integrity of surrogate pairs is skipped. This enhances performance, but restricts the range of Unicode characters that can appear in the data.

enable unicode conversion

Summary information	
Default value	0
Range of values	0 – 2
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

Activates character conversion using Unilib for the char, varchar, and text datatypes. Set enable unicode conversion to 1 to use the built-in conversion. If it cannot find a built-in conversion, Adaptive Server uses the Unilib character conversion. Set enable unicode conversion to 2 to use the appropriate Unilib conversion. Set the parameter to 0 to use only the built-in character-set conversion.

enable unicode normalization

Summary information	
Default value	1
Range of values	0 – 1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

Activates Unilib character normalization. The normalization process modifies the data so there is only a single representation in the database for a given sequence of abstract characters. Often, characters followed by combined diacritics are replaced by pre-combined forms.

Set enable unicode normalization to 1 to use the built-in process that enforces normalization on all incoming Unicode data. If this parameter is disabled (set to 0), the normalization step is bypassed and the client code is responsible for normalization rather than the server. If normalization is disabled, performance is improved—but only if *all* clients present Unicode data to the server using the same representation.

Note Once disabled, normalization cannot be turned on again. This one-way change, prevents non-normalized data from entering the data base.

size of unilib cache

Summary information	
Default value	0
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

Determines the size of the Unilib cache. size of unilib cache specifies the memory used in bytes rounded up to the nearest 1K in addition to the minimum overhead size, which provides enough memory to load a single copy of the largest Unilib conversion table. Asian clients may want to increase size of unilib cache by an extra 100K for every additional character set that they wish to support via Unicode-based conversion.

User environment

The parameters in this group configure user environments.

number of user connections

Summary information	
Default value	25
Range of values	5–2147483647
Status	Dynamic
Display level	Basic
Required role	System Administrator

number of user connections sets the maximum number of user connections that can be connected to Adaptive Server at the same time. It does not refer to the maximum number of processes; that number depends not only on the value of this parameter but also on other system activity.

Upper limit to the *maximum number of user connections*

The maximum allowable number of file descriptors per process is operating-system-dependent; see the configuration documentation for your platform.

The number of file descriptors available for Adaptive Server connections is stored in the global variable *@@max_connections*. You can report the maximum number of file descriptors your system can use with:

```
select @@max_connections
```

The return value represents the maximum number of file descriptors allowed by the system for your processes, minus overhead. Overhead increases with the number of engines. For more information on how multiprocessing affects the number file descriptors available for Adaptive Server connections, see “Managing user connections” on page 667.

In addition, you must reserve a number of connections for the following items, which you also set with configuration parameters:

- The database devices, including mirror devices
- Site handlers
- Network listeners

The following formula determines how high you can set number of user connections, number of devices, max online engines, number of remote sites, and max number network listeners:

number of user connections + (number of devices * max online engines * 2) + number of remote sites + max number network listeners cannot be greater than the value of *@@max_connections*.

Reserved connections

One connection from of the configured number of connections is reserved for temporary administrative tasks to make sure that database administrators can connect to Adaptive Server if they need to increase the number of user connections and there are no free connections. A reserved connection is allocated only to a user who has the sa_role and has a total login time of 15 minutes. After this, Adaptive Server terminates the connection to ensure the availability of the reserved connection at an installation wth multiple database administrators.

Adaptive Server uses this reserved connection automatically when a client uses the last resource for connecting to Adaptive Server.

If Adaptive Server is using a reserved connection, the following informational message is displayed when the user logs onto Adaptive Server:

```
There are not enough user connections available; you are being connected
using a temporary administrative connection which will time out after '15'
minutes. Increase the value of th 'number of user connections' parameter
```

Adaptive Server also prints a message similar to the following to the errorlog when the final connnection to Adaptive Server terminates due to a timeout:

```
00:00000:00008:2003/03/14 11:25:31.36 server  Process '16' has been
terminated as it exceeded the maximum login time allowed for such processes.
This process used a connection reserved for system administrators and has a
maximum login period of '15' minutes
```

Optimizing the value of the *max number of user connections* parameter

There is no formula for determining how many connections to allow for each user. You must estimate this number, based on the system and user requirements described here. You must also take into account that on a system with many users, there is more likelihood that connections needed only occasionally or transiently can be shared among users. The following processes require user connections:

- One connection is needed for each user running isql.
- Application developers use one connection for each editing session.

- The number of connections required by users running an application depends on how the application has been programmed. Users executing Open Client programs need one connection for each open DB-Library dbprocess or Client-Library cs_connection.

Note It is a good idea to estimate the maximum number of connections that will be used by Adaptive Server and to update number of user connections as you add physical devices or users to the system. Use sp_who periodically to determine the number of active user connections on your Adaptive Server.

Certain other configuration parameters, including stack size and default network packet size, affect the amount of memory for each user connection.

User connections for shared memory

Adaptive Server uses the value of the number of user connections parameter to establish the number of shared-memory connections for EJB Server. Thus, if number of user connections is 30, Adaptive Server establishes 10 shared-memory connections for EJB Server. Shared-memory connections are not a subset of user connections, and are not subtracted from the number of user connections.

To increase the number of user connections for shared memory, you must:

1 Increase number of user connections to a number one-third of which is the number of desired shared-memory connections.

2 Restart Adaptive Server.

Although number of user connections is a dynamic configuration parameter, you must restart the server to change the number of user connections for shared memory. See the *EJB Server User's Guide* for more information.

permission cache entries

Summary information	
Default value	15
Range of values	1–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

permission cache entries determines the number of cache protectors per task. This parameter increases the amount of memory for each user connection and worker process.

Information about user permissions is held in the permission cache. When Adaptive Server checks permissions, it looks first in the permission cache; if it does not find what it needs, it looks in the sysprotects table. It is significantly faster if Adaptive Server finds the information it needs in the permission cache and does not have to read sysprotects.

However, Adaptive Server looks in the permission cache only when it is checking user permissions, not when permissions are being granted or revoked. When a permission is granted or revoked, the entire permission cache is flushed. This is because existing permissions have timestamps that become outdated when new permissions are granted or revoked.

If users on your Adaptive Server frequently perform operations that require their permissions to be checked, you may see a small performance gain by increasing the value of permission cache entries. This effect is not likely to be significant enough to warrant extensive tuning.

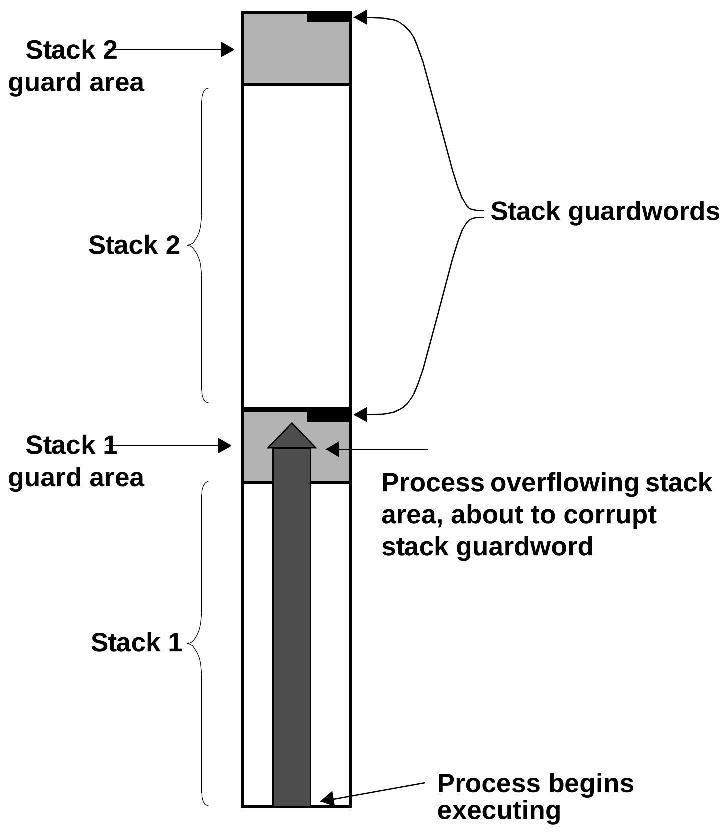
If users on your Adaptive Server frequently grant or revoke permissions, avoid setting permission cache entries to a large value. The space used for the permission cache would be wasted, since the cache is flushed with each grant and revoke command.

stack guard size

Summary information	
Default value	4096
Range of values	0–2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator

stack guard size sets the size (in bytes) of the stack guard area. The *stack guard area* is an overflow stack of configurable size at the end of each stack. Adaptive Server allocates one stack for each user connection and worker process when it starts. These stacks are located contiguously in the same area of memory, with a guard area at the end of each stack. At the end of each stack guard area is a *guardword*, which is a 4-byte structure with a known pattern. Figure 4-7 illustrates how a process can corrupt a stack guardword.

Figure 4-7: Process about to corrupt stack guardword



Adaptive Server periodically checks to see whether the stack pointer for a user connection has entered the stack guard area associated with that user connection's stack. If it has, Adaptive Server aborts the transaction, returns control to the application that generated the transaction, and generates Error 3626:

```
The transaction was aborted because it used too much
stack space.  Either use sp_configure to increase the
stack size, or break the query into smaller pieces.
spid: %d, suid: %d, hostname: %.*s, application name:
%.*s
```

Adaptive Server also periodically checks the guardword pattern to see if it has changed, thus indicating that a process has overflowed the stack boundary. When this occurs, Adaptive Server prints these messages to the error log and shuts down:

```
kernel: *** Stack overflow detected: limit: 0x%lx sp: 0x%lx
kernel: *** Stack Guardword corrupted
kernel: *** Stack corrupted, server aborting
```

In the first message, “limit” is the address of the end of the stack guard area, and “sp” is the current value of the stack pointer.

In addition, Adaptive Server periodically checks the stack pointer to see whether it is completely outside both the stack and the stack guard area for the pointer’s process. If it is, Adaptive Server shuts down, even if the guardword is not corrupted. When this happens, Adaptive Server prints the following messages to the error log:

```
kernel: *** Stack overflow detected: limit: 0x%lx sp: 0x%lx
kernel: *** Stack corrupted, server aborting
```

The default value for stack guard size is appropriate for most applications. However, if you experience server shutdown from either stack guardword corruption or stack overflow, increase stack guard size by a 2K increment. *Each* configured user connection and worker process has a stack guard area; thus, when you increase stack guard size, you use up that amount of memory, multiplied by the number of user connections and worker processes you have configured.

Rather than increasing stack guard size to avoid stack overflow problems, consider increasing stack size (see “stack size” on page 202). The stack guard area is intended as an overflow area, not as an extension to the regular stack.

Adaptive Server allocates stack space for each task by adding the values of the stack size and stack guard size parameters. stack guard size must be configured in multiples of 2K. If the value you specify is not a multiple of 2K, sp_configure verification routines round the value up to the next highest multiple.

stack size

Summary information	
Default value	platform-specific
Range of values	Platform-specific minimum–2147483647

Summary information	
Status	Static
Display level	Basic
Required role	System Administrator

stack size specifies the size (in bytes) of the execution stacks used by each user process on Adaptive Server. To find the stack size values for your platform, use sp_helpconfig or sp_configure. stack size must be configured in multiples of 2K. If the value you specify is not a multiple of 2K, sp_configure verification routines round the value up to the next highest multiple.

An *execution stack* is an area of Adaptive Server memory where user processes keep track of their process context and store local data.

Certain queries can contribute to the probability of a stack overflow. Examples include queries with extremely long where clauses, long select lists, deeply nested stored procedures, and multiple selects and updates using holdlock. When a stack overflow occurs, Adaptive Server prints an error message and rolls back the transaction. See “stack guard size” on page 200 for more information on stack overflows. See the *Troubleshooting and Error Messages Guide* for more information on specific error messages.

The two options for remedying stack overflows are to break the large queries into smaller queries and to increase stack size. Changing stack size affects the amount of memory required for *each* configured user connection and worker process. See “total logical memory” on page 145 for further information.

If you have queries that exceed the size of the execution stack, you may want to rewrite them as a series of smaller queries. This is particularly true if there are only a small number of such queries or if you run them infrequently.

There is no way to determine how much stack space a query will require without actually running the query. Stack space for each user connection and worker process is preallocated at start-up.

Therefore, determining the appropriate value for stack size is an empirical process. You should test your largest and most complex queries using the default value for stack size. If they run without generating error messages, the default is probably sufficient. If they generate error messages, you should begin by increasing stack size by a small amount (2K). Rerun your queries and see if the amount you have added is sufficient. If it is not, continue to increase stack size until queries run without generating error messages.

If you are using CIS, or if Java is enabled in the database and you want to use methods that call JDBC, Sybase recommends that you increase the default by 50 percent. If you are not using JDBC or CIS, the standard default value is sufficient.

user log cache size

Summary information	
Default value	Logical page size
Range of values	2048[a] –2147483647 a. Minimum determined by server’s logical page size
Status	Static
Display level	Intermediate
Required role	System Administrator

user log cache size specifies the size (in bytes) for each user’s log cache. It’s size is determined by the server’s logical page size. There is one user log cache for each configured user connection and worker process. Adaptive Server uses these caches to buffer the user transaction log records, which reduces the contention at the end of the transaction log.

When a user log cache becomes full or another event occurs (such as when the transaction completes), Adaptive Server “flushes” all log records from the user log cache to the database transaction log. By first consolidating the log records in each user’s log cache, rather than immediately adding each record to the database’s transaction log, Adaptive Server reduces contention of processes writing to the log, especially for SMP systems configured with more than one engine.

Note For transactions using a database with mixed data and log segments, the user log cache is flushed to the transaction log after each log record. No buffering takes place. If your databases do not have dedicated log segments, you should not increase the user log cache size.

Do not configure user log cache size to be larger than the maximum amount of log information written by an application’s transaction. Since Adaptive Server flushes the user log cache when the transaction completes, any additional memory allocated to the user log cache is wasted. If no transaction in your server generates more than 4000 bytes of transaction log records, set user log cache size no higher than that value. For example:

```
sp_configure "user log cache size", 4000
```

Setting user log cache size too high wastes memory. Setting it too low can cause the user log cache to fill up and flush more than once per transaction, increasing the contention for the transaction log. If the volume of transactions is low, the amount of contention for the transaction log may not be significant.

Use sp_sysmon to understand how this parameter affects cache behavior. See the *Performance and Tuning Guide* for more information.

user log cache spinlock ratio

Summary information	
Default value	20
Range of values	1–2147483647
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

For Adaptive Servers running with multiple engines, the user log cache spinlock ratio parameter specifies the ratio of user log caches per user log cache spinlock. There is one user log cache for each configured user connection.

The default value for this parameter is 20, or one spinlock for each 20 user connections configured for your server.

Use sp_sysmon to understand how this parameter affects cache behavior. See the *Performance and Tuning Guide* for more information.

For more information about configuring spinlock ratios, see “Configuring spinlock ratio parameters” on page 668.

CHAPTER 5

Diagnosing System Problems

This chapter discusses diagnosing and fixing system problems.

How Adaptive Server uses error messages to respond to system problems

When Adaptive Server encounters a problem, it displays information—in an error message that describes whether the problem is caused by the user or the system—about the problem, how serious it is, and what you can do to fix it. The error message consists of:

- A **message number**, which uniquely identifies the error message
- A **severity level number** between 10 and 24, which indicates the type and severity of the problem
- An **error state number**, which allows unique identification of the line of Adaptive Server code at which the error was raised
- An **error message**, which tells you what the problem is, and may suggest how to fix it

For example, this is what happens if you try to access a table that does not exist:

```
select * from publisher

Msg 208, Level 16, State 1:
publisher not found. Specify owner.objectname or use
sp_help to check whether the object exists (sp_help may
produce lots of output).
```

In some cases, there can be more than one error message for a single query. If there is more than one error in a batch or query, Adaptive Server usually reports only the first one. Subsequent errors are reported the next time you execute the batch or query.

The error messages are stored in master..sysmessages, which is updated with each new release of Adaptive Server. Here are the first few rows (from an Adaptive Server with us_english as the default language):

```
select error, severity, description
from sysmessages
where error >=101 and error <=106
and langid is null
```

```
error severity description
----- -------- --------------------------------------------------
  101       15 Line %d: SQL syntax error.
  102       15 Incorrect syntax near '%.*s'.
  103       15 The %S_MSG that starts with '%.*s' is too long.
               Maximum length is %d.
  104       15 Order-by items must appear in the select-list if
               the statement contains set operators.
  105       15 Unclosed quote before the character string '%.*s'.
  106       16 Too many table names in the query. The maximum
               allowable is %d.

(6 rows affected)
```

You can generate your own list by querying sysmessages. Here is some additional information for writing your query:

- If your server supports more than one language, sysmessages stores each message in each language. The column langid is NULL for us_english and matches the syslanguages.langid for other languages installed on the server. For information about languages on your server, use sp_helplanguage.
- The dlevel column in sysmessages is currently unused.
- The sqlstate column stores the SQLSTATE value for error conditions and exceptions defined in ANSI SQL92.

- Message numbers 17000 and greater are system procedure error messages and message strings.

Error messages and message numbers

The combination of message number (*error*) and language ID (*langid*) uniquely identifies each error message. Messages with the same message number but different language IDs are translations.

```
select error, description, langid
from sysmessages
where error = 101

error description                            langid
----- ------------------------------------ ------
  101 Line %d: SQL syntax error.              NULL
  101 Ligne %1!: erreur de syntaxe SQL.          1
  101 Zeile %1!: SQL Syntaxfehler.               2

(3 rows affected)
```

The error message text is a description of the problem. The descriptions often include a line number, a reference to a kind of database object (a table, column, stored procedure, and so forth), or the name of a particular database object.

In the description field of sysmessages, a percent sign (%) followed by a character or character string serves as a placeholder for these pieces of data, which Adaptive Server supplies when it encounters the problem and generates the error message. “%d” is a placeholder for a number; “%S_MSG” is a placeholder for a kind of database object; “%.*s”—all within quotes—is a placeholder for the name of a particular database object. Table 5-1 lists placeholders and what they represent.

For example, the description field for message number 103 is:

```
The %S_MSG that starts with '%.*s' is too long. Maximum
length is %d.
```

The actual error message as displayed to a user might be:

```
The column that starts with 'title' is too long. Maximum
length is 80.
```

For errors that you report to Technical Support, it is important that you include the numbers, object types, and object names. (See “Reporting errors” on page 218.)

Variables in error message text

Table 5-1 explains the symbols that appear in the text provided with each error message explanation:

Table 5-1: Error text symbols key

Symbol	Stands for
%d, %D	Decimal number
%x,%X,%.*x,%lx, %04x, %08lx	Hexadecimal number
%s	Null-terminated string
%.*s, %*s, %*.s	String, usually the name of a particular database object
%S_*type*	Adaptive Server-defined structure
%c	Single character
%f	Floating-point number
%ld	Long decimal
%lf	Double floating-point number

Adaptive Server error logging

Error messages from Adaptive Server are sent only to the user's screen.

The stacktrace from fatal error messages (severity levels 19 and higher) and error messages from the kernel are also sent to an error log file. The name of this file varies; see the configuration documentation for your platform or the *Utility Guide*.

Note The error log file is owned by the user who installed Adaptive Server (or the person who started Adaptive Server after an error log was removed). Permissions or ownership problems with the error log at the operating system level can block successful start-up of Adaptive Server.

Adapitve Server creates an error log for you if one does not already exist. You specify the location of the error log at start-up with the *errorlogfile* parameter in the runserver file or at the command line. The Sybase installation utility configures the runserver file with *$SYBASE/install* as the location of the error log if you do not choose an alternate location during installation. If you do not specify the location in the runserver file or at the command line, the location of the error log is the directory from which you start Adaptive Server. For more information about specifying the location of the error log, see description of the dataserver command in the *Utility Guide*.

Note Always start Adaptive Server from the same directory, or with the runserver file or the error log flag, so that you can locate your error log.

Each time you start a server, messages in the error log provide information on the success (or failure) of the start and the recovery of each database on the server. Subsequent fatal error messages and all kernel error messages are appended to the error log file. If you need to reduce the size of the error log by deleting old or unneeded messages, you must “prune” the log while Adaptive Server is shut down.

Error log format

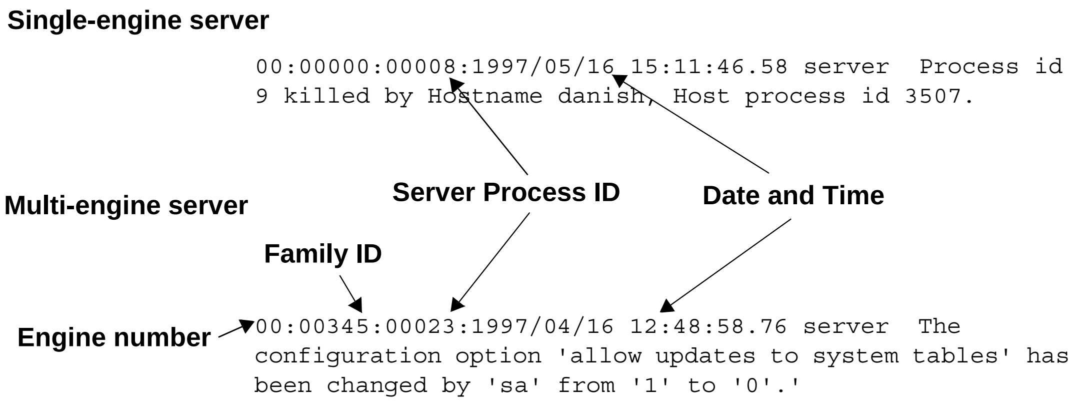
Entries in the error log include the following information:

- The engine involved for each log entry. The engine number is indicated by a 2-digit number. If only one engine is online, the display is “00.”
- The family ID of the originating thread:
    - In serial processing, the display is “00000.”
    - In parallel processing, the display is the server process ID number of the parent of the originating thread.
- The server process ID of the originating thread:
    - In serial processing, this is the server process ID number of the thread that generated the message. If the thread is a system task, then the display is “00000.”
    - In parallel processing, this is the server process ID number of the originating thread.

- The date, displayed in the format `yyyy/mm/dd`, which allows you to sort error messages by date.
- The time, displayed in 24-hour format, which includes seconds and hundredths of a second.
- The word “server” or “kernel.” This entry is for Sybase Technical Support use only.
- The error message itself.

Figure 5-1 shows two examples of a line from an error log:

Figure 5-1: Error log format



```
00:00000:00008:1997/05/16 15:11:46.58 server  Process id
9 killed by Hostname danish, Host process id 3507.
```

```
00:00345:00023:1997/04/16 12:48:58.76 server  The
configuration option 'allow updates to system tables' has
been changed by 'sa' from '1' to '0'.'
```

Severity levels

The severity level of a message indicates information about the type and severity of the problem that Adaptive Server has encountered. For maximum integrity, when Adaptive Server responds to error conditions, it displays messages from sysmessages, but takes action according to an internal table. A few corresponding messages differ in severity levels, so you may occasionally notice a difference in expected behavior if you are developing applications or procedures that refer to Adaptive Server messages and severity levels.

Warning! You can create your own error numbers and messages based on Adaptive Server error numbers (for example, by adding 20,000 to the Adaptive Server value). However, you cannot alter the Adaptive Server-supplied system messages in the sysmessages system table.

You can add user-defined error messages to sysusermessages with sp_addmessage. See the *Reference Manual*.

Users should inform the System Administrator whenever problems that generate severity levels of 17 and higher occur. The System Administrator is responsible for resolving them and tracking their frequency.

If the problem has affected an entire database, the System Administrator may have to use the database consistency checker (dbcc) to determine the extent of the damage. The dbcc may identify some objects that have to be removed. It can repair some damage, but the database may have to be reloaded.

For more information, refer to the following chapters:

- dbcc is discussed in Chapter 26, “Checking Database Consistency.”
- Loading a user database is discussed in Chapter 28, “Backing Up and Restoring User Databases.”
- Loading system databases is discussed in Chapter 29, “Restoring the System Databases.”

The following sections discuss each severity level.

Levels 10–18

Error messages with severity levels 10–16 are generated by problems that are caused by user errors. These problems can always be corrected by the user. Severity levels 17 and 18 do not terminate the user’s session.

Error messages with severity levels 17 and higher should be reported to the System Administrator or Database Owner.

Level 10: Status information

Messages with severity level 10 are not errors at all. They provide additional information after certain commands have been executed and, typically, do not display the message number or severity level. For example, after a create database command has been run, Adaptive Server displays a message telling the user how much of the requested space has been allocated for the new database.

Level 11: Specified database object not found

Messages with severity level 11 indicate that Adaptive Server cannot find an object that was referenced in the command.

This is often because the user has made a mistake in typing the name of a database object, because the user did not specify the object owner's name, or because of confusion about which database is current. Check the spelling of the object names, use the owner names if the object is not owned by you or "dbo," and make sure you are in the correct database.

Level 12: Wrong datatype encountered

Messages with severity level 12 indicate a problem with datatypes. For example, the user may have tried to enter a value of the wrong datatype in a column or to compare columns of different and incompatible datatypes.

To correct comparison problems, use the convert function with select. For information on convert, see the *Reference Manual* or the *Transact-SQL User's Guide*.

Level 13: User transaction syntax error

Messages with severity level 13 indicate that something is wrong with the current user-defined transaction. For example, you may have issued a commit transaction command without having issued a begin transaction or you may have tried to roll back a transaction to a savepoint that has not been defined (sometimes there may be a typing or spelling mistake in the name of the savepoint).

Severity level 13 can also indicate a deadlock, in which case the deadlock victim's process is rolled back. The user must restart his or her command.

Level 14: Insufficient permission to execute command

Messages with severity level 14 mean that you do not have the necessary permission to execute the command or access the database object. You can ask the owner of the database object, the owner of the database, or the System Administrator to grant you permission to use the command or object in question.

Level 15: Syntax error in SQL statement

Messages with severity level 15 indicate that the user has made a mistake in the syntax of the command. The text of these error messages includes the line numbers on which the mistake occurs and the specific word near which it occurs.

Level 16: Miscellaneous user error

Most error messages with severity level 16 reflect that the user has made a nonfatal mistake that does not fall into any of the other categories. Severity level 16 and higher can also indicate software or hardware errors.

For example, the user may have tried to update a view in a way that violates the restrictions. Another error that falls into this category is unqualified column names in a command that includes more than one table with that column name. Adaptive Server has no way to determine which one the user intends. Check the command syntax and working database context.

Messages that ordinarily have severities greater than 16 will show severity 16 when they are raised by dbcc checktable or dbcc checkalloc so that checks can continue to the next object. When you are running the dbcc utility, check the *Error Messages and Troubleshooting Guide* for information about error messages between 2500 and 2599 with a severity level of 16.

Note Levels 17 and 18 are usually not reported in the error log. Users should be instructed to notify the System Administrator when level 17 and 18 errors occur.

Level 17: Insufficient resources

Error messages with severity level 17 mean that the command has caused Adaptive Server to run out of resources or to exceed some limit set by the System Administrator. You can continue with the work you are doing, although you may not be able to execute a particular command.

These system limits include the number of databases that can be open at the same time and the number of connections allowed to Adaptive Server. They are stored in system tables and can be checked with sp_configure. See Chapter 4, “Setting Configuration Parameters,” for more information on changing configuration variables.

The Database Owner can correct the level 17 error messages indicating that you have run out of space. Other level 17 error messages should be corrected by the System Administrator.

Level 18: Non-fatal internal error detected

Error messages with severity level 18 indicate some kind of internal software bug. However, the command runs to completion, and the connection to Adaptive Server is maintained. You can continue with the work you are doing, although you may not be able to execute a particular command. An example of a situation that generates severity level 18 is Adaptive Server detecting that a decision about the access path for a particular query has been made without a valid reason.

Since problems that generate such messages do not keep users from their work, users tend not to report them. Users should be instructed to inform the System Administrator every time an error message with this severity level (or higher) occurs so that the System Administrator can report them.

Severity levels 19–26

Fatal problems generate error messages with severity levels 19 and higher. They break the user’s connection to Adaptive Server (some of the higher severity levels shut down Adaptive Server). To continue working, the user must restart the client program.

When a fatal error occurs, the process freezes its state before it stops, recording information about what was happening. It is then killed and disappears.

When the user’s connection is broken, he or she may or may not be able to reconnect and resume working. Some problems with severity levels in this range affect only one user and one process. Others affect all the processes in the database. In some cases, it will be necessary to restart Adaptive Server. These problems do not necessarily damage a database or its objects, but they can. They may also result from earlier damage to a database or its objects. Other problems are caused by hardware malfunctions.

A backtrace of fatal error messages from the kernel is directed to the error log file, where the System Administrator can review it.

Level 19: Adaptive Server fatal error in resource

Error messages with severity level 19 indicate that some non-configurable internal limit has been exceeded and that Adaptive Server cannot recover gracefully. You must reconnect to Adaptive Server.

Level 20: Adaptive Server fatal error in current process

Error messages with severity level 20 indicate that Adaptive Server has encountered a bug in a command. The problem has affected only the current process, and it is unlikely that the database itself has been damaged. Run dbcc diagnostics. You must reconnect to Adaptive Server.

Level 21: Adaptive Server fatal error in database processes

Error messages with severity level 21 indicate that Adaptive Server has encountered a bug that affects all the processes in the current database. However, it is unlikely that the database itself has been damaged. Restart Adaptive Server and run the dbcc diagnostics. You must reconnect to Adaptive Server.

Level 22: Adaptive Server fatal error: Table integrity suspect

Error messages with severity level 22 indicate that the table or index specified in the message was previously damaged by a software or hardware problem.

The first step is to restart Adaptive Server and run dbcc to determine whether other objects in the database are also damaged. Whatever the report from dbcc may be, it is possible that the problem is in the cache only and not on the disk itself. If so, restarting Adaptive Server will fix the problem.

If restarting does not help, then the problem is on the disk as well. Sometimes, the problem can be solved by dropping the object specified in the error message. For example, if the message tells you that Adaptive Server has found a row with length 0 in a nonclustered index, the table owner can drop the index and re-create it.

Adaptive Server takes any pages or indexes offline that it finds to be suspect during recovery. Use sp_setsuspect_granularity to determine whether recovery marks an entire database or only individual pages as suspect. See sp_setsuspect_granularity in the *Reference Manual* for more information.

You must reconnect to Adaptive Server.

Level 24: Hardware error or system table corruption

Error messages with severity level 24 reflect some kind of media failure or (in rare cases) the corruption of sysusages. The System Administrator may have to reload the database. It may be necessary to call your hardware vendor.

Level 23: Fatal error: Database integrity suspect

Error messages with severity level 23 indicate that the integrity of the entire database is suspect due to previous damage caused by a software or hardware problem. Restart Adaptive Server and run dbcc diagnostics.

Even when a level 23 error indicates that the entire database is suspect, the damage may be confined to the cache, and the disk itself may be fine. If so, restarting Adaptive Server with startserver will fix the problem.

Level 25: Adaptive Server internal error

Level 25 errors are not displayed to the user; this level is only used for Adaptive Server internal errors.

Level 26: Rule error

Error messages with severity level 26 reflect that an internal locking or synchronization rule was broken. You must shut down and restart Adaptive Server.

Reporting errors

When you report an error, include:

- The message number, level number, and state number.
- Any numbers, database object types, or database object names that are included in the error message.
- The context in which the message was generated, that is, which command was running at the time. You can help by providing a hard copy of the backtrace from the error log.

Backup Server error logging

Like Adaptive Server, Backup Server creates an error log if one does not already exist.You specify the location of the error log at start-up with the *error_log_file* parameter in the runserver file or at the command line. The Sybase installation utility configures the runserver file with *$SYBASE/install* as the location of the error log if +you do not choose an alternate location during installation. If you do not specify the location in the runserver file or at the command line, the location of the error log is the directory from which you start Backup Server. Use the backupserver -V option (bcksvr -V on Windows NT) to limit the messages printed to the error log. For more information about specifying the location of the error log, see the sections describing Backup Server in the *Utility Guide*.

Backup Server error messages are in the form:

```
MMM DD YYY: Backup Server:N.N.N.N: Message Text
```

Backup Server message numbers consist of 4 integers separated by periods, in the form N.N.N.N. Messages in the form N.N.N are sent by Open Server™.

The four components of a Backup Server error message are *major.minor.severity.state*:

- The *major* component generally indicates the functional area of the Backup Server code where the error occurred:
  - 1 – System errors
  - 2 – Open Server event errors
  - 3 – Backup Server remote procedure call errors
  - 4 – I/O service layer errors
  - 5 – Network data transfer errors
  - 6 – Volume handling errors
  - 7 – Option parsing errors

  Major error categories 1– 6 may result from Backup Server internal errors or a variety of system problems. Major errors in category 7 are almost always due to problems in the options you specified in your dump or load command.
- *minor* numbers are assigned in order within a major category.
- *severity* is:
  - 1 – Informational, no user action necessary.

    - 2, 3 – An unexpected condition, possibly fatal to the session, has occurred. The error may have occurred with usage, environment, or internal logic, or any combination of these factors.
    - 4 – An unexpected condition, fatal to the execution of the Backup Server, has occurred. The Backup Server must exit immediately.
- *state* codes have a one-to-one mapping to instances of the error report within the code. If you need to contact Technical Support about Backup Server errors, the state code helps determine the exact cause of the error.

Killing processes

A process is a unit of execution carried out by Adaptive Server. Each process is assigned a unique process identification number when it starts, this number is called a spid. These numbers are stored, along with other information about each process, in master..sysprocesses. Processes running in a parallel processes environment create child processes, each of which has its own spids. Several processes create and assign spids: starting Adaptive Server, login tasks, checkpoints, the housekeeper tasks, and so on. You can see most of the information by running sp_who.

Running sp_who on a single-engine server shows the sp_who process running and all other processes that are "runnable" or in one of the sleep states. In multi-engine servers, there can be a process running for each engine.

The kill command gets rid of an ongoing process. The most frequent reason for killing a process is that it interferes with other users and the person responsible for running it is not available. The process may hold locks that block access to database objects, or there may be many sleeping processes occupying the available user connections. A System Administrator can kill processes that are:

- Waiting for an alarm, such as a waitfor command
- Waiting for network sends or receives
- Waiting for a lock
- Waiting for synchronization messages from another process in a family
- Most running or "runnable" processes

Adaptive Server allows you to kill processes only if it can cleanly roll back any uncompleted transactions and release all system resources that are used by the process. For processes that are part of a family, killing any of the child processes will also kill all other processes in the family. However, it is easiest to kill the parent process. For a family of processes, the kill command is detected more quickly if the status of the child processes is `sync sleep`.

Table 5-2 shows the values that sp_who reports and when the kill command takes effect.

Table 5-2: Status values reported by sp_who

Status	Indicates	Effects of kill command
`recv sleep`	Waiting on a network read	Immediate.
`send sleep`	Waiting on a network send	Immediate.
`alarm sleep`	Waiting on an alarm such as: `waitfor delay "10:00"`	Immediate.
`lock sleep`	Waiting on a lock acquisition	Immediate.
`sync sleep`	Waiting on a synchronization message from another process in the family.	Immediate. Other processes in the family must also be brought to state in which they can be killed.
`sleeping`	Waiting on a disk I/O, or some other resource. Probably indicates a process that is running, but doing extensive disk I/O	Killed when it “wakes up,” usually immediate; a few sleeping processes do not wake up and require a Server restart to clear.
`runnable`	In the queue of runnable processes	Immediate.
`running`	Actively running on one of the server engines	Immediate.
`infected`	Server has detected serious error condition; extremely rare	kill command not recommended. Server restart probably required to clear process.
`background`	A process, such as a threshold procedure, run by Adaptive Server rather than by a user process	Immediate; use kill with extreme care. Recommend a careful check of sysprocesses before killing a background process.
`log suspend`	Processes suspended by reaching the last-chance threshold on the log	Immediate.

Only a System Administrator can issue the kill command; permission to use it cannot be transferred.

The syntax is:

kill *spid*

You can kill only one process at a time, but you can perform a series of kill commands in a batch. For example:

```
1> kill 7
```

```
2> kill 8
3> kill 9
4> go
```

A kill command is not reversible and cannot be included in a user-defined transaction. spid must be a numeric constant; you cannot use a variable. Here is some sample output from sp_who:

```
fid spid status     loginame origname hostname  blk  dbname cmd
--- ---- ---------  -------- -------- --------  ---  ------ ----------------
0   1   recv sleep  howard   howard   svr30eng  0    master AWAITING COMMAND
0   2   sleeping    NULL     NULL               0    master NETWORK HANDLER
0   3   sleeping    NULL     NULL               0    master DEADLOCK TUNE
0   4   sleeping    NULL     NULL               0    master MIRROR HANDLER
0   5   sleeping    NULL     NULL               0    master CHECKPOINT SLEEP
0   6   sleeping    NULL     NULL               0    master HOUSEKEEPER
0   7   recv sleep  bill     bill     bigblue   0    master AWAITING COMMAND
0   8   recv sleep  wilbur   wilbur   hazel     0    master AWAITING COMMAND
0   9   recv sleep  joan     joan     luv2work  0    master AWAITING COMMAND
0   10  running     foote    foote    svr47hum  0    master SELECT
(10 rows affected, return status = 0)
```

In the example above, processes 2–6 cannot be killed: they are system processes. The login name NULL and the lack of a host name identify them as system processes. You will always see NETWORK HANDLER, MIRROR HANDLER, HOUSEKEEPER, and CHECKPOINT SLEEP (or, rarely, CHECKPOINT). AUDIT PROCESS becomes activated if you enable auditing.

Processes 1, 8, 9, and 10 can be killed, since they have the status values "recv sleep," "send sleep," "alarm sleep," and "lock sleep."

In sp_who output, you cannot tell whether a process whose status is "recv sleep" belongs to a user who is using Adaptive Server and may be pausing to examine the results of a command or whether the process indicates that a user has restarted a PC or other terminal, and left a stranded process.You can learn more about a questionable process by querying the sysprocesses table for information. For example, this query shows the host process ID and client software used by process 8:

```
select hostprocess, program_name
    from sysprocesses
where spid = 8

hostprocess program_name
----------- ----------------
3993        isql
```

This query, plus the information about the user and host from the sp_who results, provides additional information for tracking down the process from the operating system level.

Using *sp_lock* to examine blocking processes

In addition to sp_who, sp_lock can help identify processes that are blocking other processes. If the blk column in the sp_who report indicates that another process has been blocked while waiting to acquire locks, sp_lock can display information about the blocking process. For example, process 10 in the sp_who output above is blocked by process 7. To see information about process 7, execute:

```
sp_lock 7
```

For more information about locking in Adaptive Server, see the *Performance and Tuning Guide*.

Housekeeper functionality

The housekeeper provides important functionalities:

- the general automatic restart of housekeeper-related system tasks: you need not restart the server if these system tasks quit unexpectedly.
- The housekeeper feature consists of three tasks: housekeeper wash, housekeeper garbage collection, and housekeeper chores. sp_who recognizes all three tasks, as the following output shows:

```
 fid    spid   status       loginame     origname     hostname   blk_sp
id
         dbname     cmd             block_xloid

0      8 sleeping     NULL         NULL                     0

         master     HK WASH                    0

0      9 sleeping     NULL         NULL                     0

         master     HK GC                      0

0     10 sleeping     NULL         NULL                     0
```

```
          master     HK CHORES                  0
0      12 recv sleep    sa            sa            chaucer    0

(11 rows affected, return status = 0)
```

- A System Administrator can change all housekeeper task priorities.

  sp_showpsexe, as well as sp_who, recognizes all three housekeeper names.

For more information about sp_who and sp_showpsexe, see the *Reference Manual*.

Three housekeepers

The housekeeper work is divided among three separate tasks:

- Housekeeper wash task
- Housekeeper chores task
- Housekeeper garbage collection task

The output for all three tasks appears in the output for sp_who.

Housekeeper wash

Washing buffers is an optional task and runs at idle times only. You can turn off this task using the configuration parameter housekeeper free write percent. The housekeeper wash task is the only housekeeper task for which you use this configuration parameter.

Housekeeper chores

The housekeeper chores task runs at idle times only and does not use a common configuration parameter. It manages miscellaneous chores, such as:

- Flushing table statistics.
- Flushing account statistics.
- Handling timeout of detached transactions. You can turn off this chore using the configuration parameter dtm detach timeout period.

- Checking licence usage. You can turn this task off using the configuration parameter license information.

Housekeeper garbage collection

There are two forms of garbage collection, lazy and aggressive. These terms describe two distinct tests for finding empty pages.

- Lazy garbage collection refers to an inexpensive test to find empty pages. This test may not be effective during long-running transactions, and empty pages may accumulate. Lazy garbage collection is inexpensive to use, but can lower performance. Performance is affected by the fragmentation of space allocated to a table, and by the accumulation of empty pages that must be evaluated during queries.
- Aggressive garbage collection refers to a sophisticated test for empty pages. This test is more expensive than the lazy garbage collection test, because it checks each deleted row in a page to determine whether that deleting transactions are committed.

  Both the delete command and the housekeeper garbage collection task can be configured for aggressive or lazy garbage collection, through the configuration parameter enable housekeeper GC.

  The aggressive housekeeper garbage collection self-tunes the frequency with which the housekeeper garbage collection task examines the housekeeper list, so that the frequency of examination matches the rate at which the application generates empty pages.

Running at user priority

The housekeeper garbage collection task operates at the priority level of an ordinary user, competing for CPU time with ordinary user tasks. This behavior prevents the list of empty pages from growing faster than the housekeeper can delete them.

Configuring enable housekeeper GC

To configure Adaptive Server for garbage collection task, use:

```
sp_configure "enable housekeeper GC", value
```

For example, enter:

```
sp_configure "enable housekeeper GC", 4
```

The following are the valid values for enable housekeeper GC configuration parameter:

- 0 – disables the housekeeper garbage collection task, but enables lazy garbage collection by the delete command. You must use reorg reclaim_space to deallocate empty pages. This is the cheapest option with the lowest performance impact, but it may cause performance problems if many empty pages accumulate. Sybase does not recommend using this value.
- 1 – enables lazy garbage collection, by both the housekeeper garbage collection task and the delete command. This is the default value. If more empty pages accumulate than your application allows, consider options 4 or 5. You can use the optdiag utility to obtain statistics of empty pages.
- 2 – reserved for future use.
- 3 – reserved for future use.
- 4 – enables aggressive garbage collection for both the housekeeper garbage collection task and the delete command. This option is the most effective, but the delete command is the most expensive. This option is ideal if the deletes on your DOL tables are in a batch.
- 5 – enables aggressive garbage collection for the housekeeper, and lazy garbage collection by delete. This option is less expensive for deletes than option 4. This option is suitable when deletes are caused by concurrent transactions.

Using the reorg command

Garbage collection is most effective when you set enable housekeeper GC to 4 or 5. Sybase recommends that you set the parameter value to 5. However, if performance considerations prevent setting this parameter to 4 or 5, and you have an accumulation of empty pages, run reorg on the affected tables. You can obtain statistics on empty pages through the optdiag utility.

When the server is shut down or crashes, requests to deallocate pages that the housekeeper garbage collection task has not yet serviced are lost. These pages, empty but not deallocated by the housekeeper garbage collection task remain allocated until you remove them by running reorg.

See Chapter 25, “Using the reorg Command” for more information on running reorg.

Configuring Adaptive Server to save SQL batch text

Occasionally a query or procedure causes Adaptive Server Monitor to hang. Users with the System Administrator role can configure Adaptive Server to give Adaptive Server Monitor access to the text of the currently executing SQL batch. Viewing the SQL text of long-running batches helps you debug hung processes or fine-tune long statements that are heavy resource consumers.

Adaptive Server must be configured to collect the SQL batch text and write it to shared memory, where the text can be read by Adaptive Server Monitor Server (the server component of Adaptive Server Monitor). The client requests might come from Monitor Viewer, which is a plug-in to Sybase Central, or other Adaptive Server Monitor Server applications.

Configuring Adaptive Server to save SQL batch text also allows you to view the current query plan in showplan format (as you would see after setting showplan on). You can view the current query plan from within Adaptive Server; see "Viewing the query plan of a SQL statement" on page 230. SQL batches are viewable only through Adaptive Server Monitor Server. See the Adaptive Server Monitor Server documentation for more information about displaying the batch text.

Because the query or procedure you are viewing may be nested within a batch of SQL text, the sysprocesses table now includes columns for the line number, statement number and spid of a hung statement to view its query plan.

By default, Adaptive Server is not configured to save SQL batch text, so you must configure Adaptive Server to allocate memory for this feature. Adaptive Server Monitor access to SQL has no effect on performance if you have not configured any memory to save SQL batches.

Allocating memory for batch text

You can configure the amount of the SQL text batch you want to save. When text saving is enabled, Adaptive Server copies the subsequent SQL text batches to memory shared with SQL Server Monitor. Because each new batch clears the memory for the connection and overwrites the previous batch, you can view only currently executing SQL statements.

v **Saving SQL text**

1 Configure the amount of SQL text retained in memory (see "Configuring the amount of SQL text retained in memory" on page 228).

2 Enable SQL Server to start saving SQL text (see “Enabling Adaptive Server to start saving SQL text” on page 229).

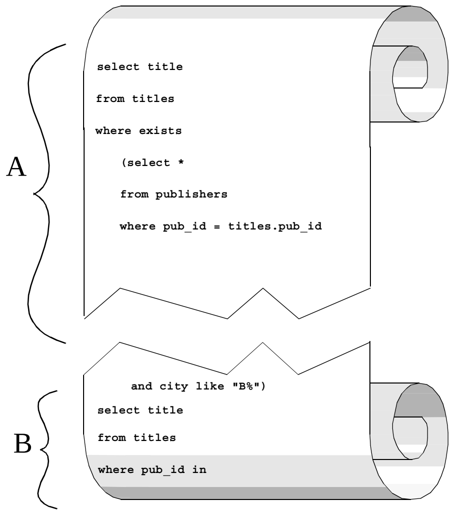
Note You must have System Administration privileges to configure and save SQL text batches.

Configuring the amount of SQL text retained in memory

After installation, you must decide the maximum amount of SQL text that can be copied to shared memory. Consider the following to help you determine how much memory to allocate per user:

- SQL batches exceeding the allocated amount of memory are truncated without warning. If you do not allocate enough memory for the batch statements, the text you are interested in viewing might be the section of the batch that is truncated, as illustrated in Figure 5-2.

Figure 5-2: How SQL text is truncated if not enough memory is configured



For example, if you configure Adaptive Server to save the amount of text designated by bracket A in the illustration, but the statement that is running occurs in the text designated by bracket B, Adaptive Server will not display the statement that is running.

- The more memory you allocate for SQL text from shared memory, the less chance the problem statement will be truncated from the batch copied to shared memory. However, Adaptive Server immediately rejects very large values because they do not leave enough memory for data and procedure caches.

Sybase recommends an initial value of 1024 bytes per user connection.

Use sp_configure with the max SQL text monitored configuration parameter to allocate shared memory, where *bytes_per_connection* (the maximum number of bytes saved for each client connection) is between 0 (the default) and 2,147,483,647 (the theoretical limit):

```
sp_configure "max SQL text monitored", bytes_per_connection
```

Since memory for SQL text is allocated by Adaptive Server at start-up, you must restart Adaptive Server for this parameter to take effect.

The total memory allocated for the SQL text from shared memory is the product of *bytes_per_connection* multiplied by the number of user connections.

Enabling Adaptive Server to start saving SQL text

After you allocate shared memory for SQL text, Adaptive Server saves a copy of each SQL batch whenever you enable an Adaptive Server Monitor event summary that includes SQL batches.

You may also have to reconfigure Adaptive Server Monitor's event buffer scan interval for SQL text. See the Adaptive Server Monitor documentation for more information.

SQL commands not represented by text

If you use Client-Library™ functions not represented by text (such as ct_cursor or ct_dynamic) to issue SQL commands, Client-Library encodes the information for efficiency, and Adaptive Server generally decodes and displays key command information. For example, if you open a cursor with ct_cursor and the command is running, the Adaptive Server Monitor event summary displays the cursor name and the cursor declare statement.

Table 5-3 lists a complete list of the Client-Library functions not represented by text:

Table 5-3: SQL commands not represented by text

Client-Library routine	DB-Library routine	Presentation name	Presentation data
ct_cursor	N/A	CLOSE_CURSOR	Cursor name, statement
ct_cursor	N/A	DECLARE_CURSOR	Cursor name, statement
ct_cursor	N/A	DELETE_AT_CURSOR	Cursor name, statement
ct_cursor	N/A	FETCH_CURSOR	Cursor name, statement
ct_fetch (when processing the results of ct_cursor)	N/A	FETCH_CURSOR	Cursor name, statement
ct_cursor CURSOR_ROWS, or ct_cancel when the connection has Client-Library cursors	N/A	CURSOR_INFO	Cursor name, statement
ct_cursor	N/A	OPEN_CURSOR	Cursor name, statement
ct_cursor	N/A	UPDATE_AT_CURSOR	Cursor name, statement
ct_command (CS_RPC_CMD) (default behavior)	dbrpcinit (only in version 10.0.1 or later)	DBLIB_RPC	RPC name
ct_dynamic	N/A	DYNAMIC_SQL	Dynamic statement name, statement
ct_command (CS_MSG_CMD	N/A	MESSAGE	None
ct_param	dbrpcparam	PARAM_FORMAT	None
ct_param	dbrpcparam	PARAMS	None
ct_command (CS_RPC_CMD) (only when a TDS version earlier than 5.0 is used)	dbrpcparam (in DB-Library version earlier than 10.0.1)	RPC	RPC name

For more information about SQL commands not represented by text, see your Open Client documentation.

Viewing the query plan of a SQL statement

Use sp_showplan and the *spid* of the user connection in question to retrieve the query plan for the statement currently running on this connection. You can also use sp_showplan to view the query plan for a previous statement in the same batch.

The syntax is:

```
declare @batch int
declare @context int
declare @statement int
execute sp_showplan <spid_value>, @batch_id= @batch output,
@context_id= @context output, @stmt_num=@statement output
```

where:

- *batch_id* – is the unique number for a batch
- *context_id* – is a unique number for every procedure (or trigger) executed in the batch
- *stmt_num* – is the number of the current statement within a batch

Adaptive Server uses the unique batch ID to synchronize the query plan with the batch text and other data retrieved by Adaptive Server Monitor.

Note You must be a System Administrator to execute sp_showplan.

For example, to see the query plan for the current statement for spid 99:

```
declare @batch int
declare @context int
declare @statement int
exec sp_showplan 99, @batch output, @context output, @statement output
```

You can run the query plan procedure independently of Adaptive Server Monitor, regardless of whether or not Adaptive Server has allocated shared memory for SQL text.

Viewing previous statements

To see the query plan for the previous statement in the same batch, issue sp_showplan with the same values as the original query, but subtract one from the statement number. Using this method, you can view all the statements in the statement batch back to query number one.

Viewing a nested procedure

Although sp_showplan allows you to view the query plan for the current statement, the actual statement that is running may exist within a procedure (or within a nested chain of procedures) called from the original SQL batch. Table 5-4 shows the columns in sysprocesses that contain information about these nested statements.

Table 5-4: Columns added to sysprocesses

Column	Datatype	Specifies
id	Integer	The object ID of the running procedure (or 0 if no procedure is running)
stmtnum	Integer	The current statement number within the running procedure (or the SQL batch statement number if no procedure is running)
linenum	Integer	The line number of the current statement within the running stored procedure (or the line number of the current SQL batch statement if no procedure is running)

This information is saved in sysprocesses, regardless of whether SQL text is enabled or any memory is allocated for SQL text.

To display the id, stmtnum, and linenum columns, enter:

```
select id, stmtnum, linenum
from sysprocesses
where spid = spid_of_hung_session
```

Note You do not need the sa_role to run this select statement.

Shutting down servers

A System Administrator can shut down Adaptive Server or Backup Server with the shutdown command. The syntax is:

shutdown [*backup_server_name*] [with {wait|nowait}]

The default for the shutdown command is with wait. That is, shutdown and shutdown with wait do exactly the same thing.

Shutting down Adaptive Server

If you do not give a server name, shutdown shuts down the Adaptive Server you are using. When you issue a shutdown command, Adaptive Server:

1 Disables logins, except for System Administrators
2 Performs a checkpoint in each database, flushing pages that have changed from memory to disk
3 Waits for currently executing SQL statements or procedures to finish

In this way, shutdown minimizes the amount of work that automatic recovery must do when you restart Adaptive Server.

The with nowait option shuts down Adaptive Server immediately. User processes are aborted, and recovery may take longer after a shutdown with nowait. You can help minimize recovery time by issuing a checkpoint command before you issue a shutdown with nowait command.

Shutting down a Backup Server

To shut down a Backup Server, give the Backup Server's name:

```
shutdown SYB_BACKUP
```

The default is with wait, so any dumps or loads in progress will complete before the Backup Server process halts. After you issue a shutdown command, no new dump or load sessions can be started on the Backup Server.

To see the names of the Backup Servers that are accessible from your Adaptive Server, execute sp_helpserver. Use the value in the name column in the shutdown command. You can shut down a Backup Server only if it is:

- Listed in sysservers on your Adaptive Server, and
- Listed in your local *interfaces* file.

Use sp_addserver to add a Backup Server to sysservers.

Checking for active dumps and loads

To see the activity on your Backup Server before executing a shutdown command, run sp_who on the Backup Server:

```
SYB_BACKUP...sp_who
```

```
spid   status   loginame hostname   blk cmd
```

```
------ -------- -------- ---------- --- ----------------
     1 sleeping NULL     NULL       0   CONNECT HANDLER
     2 sleeping NULL     NULL       0   DEFERRED HANDLER
     3 runnable NULL     NULL       0   SCHEDULER
     4 runnable NULL     NULL       0   SITE HANDLER
     5 running  sa       heliotrope 0   NULL
```

Using *nowait* on a Backup Server

The shutdown *backup_server* with nowait command shuts down the Backup Server, regardless of current activity. Use it only in severe circumstances. It can leave your dumps or loads in incomplete or inconsistent states.

If you use shutdown with nowait during a log or database dump, check for the message indicating that the dump completed. If you did not receive this message, or if you are not sure whether the dump completed, your next dump should be a dump database, not a transaction dump. This guarantees that you will not be relying on possibly inconsistent dumps.

If you use shutdown with nowait during a load of any kind, and you did not receive the message indicating that the load completed, you may not be able to issue further load transaction commands on the database. Be sure to run a full database consistency check (dbcc) on the database before you use it. You may have to reissue the full set of load commands, starting with load database.

Learning about known problems

The release bulletin is a valuable resource for learning about known problems or incompatibilities with Adaptive Server and Backup Server. Reading the release bulletin in advance can save you the time and guesswork of troubleshooting known problems.

The Adaptive Server installation program also installs files that list all system problem reports (SPRs) and closed problem reports (CPRs) for Adaptive Server. Problem reports are organized by functional areas of the product. For example, a file named *cpr_bus* would contain a listing of closed (fixed) problem reports pertaining to the Backup Server, and the file *spr_bus* would contain a list of currently open problem reports for the Backup Server.

See the release bulletin to learn the location of CPR and SPR files.

CHAPTER 6

Limiting Access to Server Resources

This chapter describes how to use resource limits to restrict the I/O cost, row count, or processing time that an individual login or application can use during critical times. It also describes how to create named time ranges to specify contiguous blocks of time for resource limits.

What are resource limits?

Adaptive Server provides resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. A *resource limit* is a set of parameters specified by a System Administrator to prevent an individual login or application from:

- Exceeding estimated or actual I/O costs, as determined by the optimizer
- Returning more than a set number of rows
- Exceeding a given elapsed time

The set of parameters for a resource limit includes the time of day to enforce the limit and the type of action to take. For example, you can prevent huge reports from running during critical times of the day, or kill a session whose query produces unwanted **Cartesian products**.

Planning resource limits

In planning a resource limit, consider:

- When to impose the limit (times of day and days of the week)
- Which users and applications to monitor
- What type of limit to impose
    - I/O cost (estimated or actual) for queries that may require large numbers of logical and physical reads
    - Row count for queries that may return large result sets
    - Elapsed time for queries that may take a long time to complete either because of their own complexity or because of external factors such as server load
- Whether to apply a limit to individual queries or to specify a broader scope (query batch or transaction)
- Whether to enforce the I/O cost limits prior to or during execution
- What action to take when the limit is exceeded (issue a warning, abort the query batch or transaction, or kill the session)

After completing the planning, use system procedures to:

- Specify times for imposing the limit by creating a named time range using sp_add_time_range
- Create new resource limits using sp_add_resource_limit
- Obtain information about existing resource limits using sp_help_resource_limit
- Modify time ranges and resource limits using sp_modify_time_range and sp_modify_resource_limit, respectively
- Drop time ranges and resource limits using sp_drop_time_range and sp_drop_resource_limit, respectively

Enabling resource limits

Configure Adaptive Server to enable resource limits. Use allow resource limits configuration parameter:

```
sp_configure "allow resource limits", 1
```

1 enables the resource limits; 0 disables them. allow resource limits is static, so you must restart the server to reset the changes.

allow resource limits signals the server to allocate internal memory for time ranges, resource limits, and internal server alarms. It also internally assigns applicable ranges and limits to login sessions.

Setting allow resource limits to 1 also changes the output of showplan and statistics i/o, as follows:

- showplan displays estimated I/O cost information for DML statements. The information displayed is the optimizer's cost estimate for the query as a unitless number. The total estimated I/O cost is displayed for the query as a whole. This cost estimate is dependent on the table statistics (number and distribution of values) and the size of the appropriate buffer pools. It is independent of such factors as the state of the buffer pools and the number of active users. For more information, see "Messages describing access methods, caching, and I/O cost" on page 93 in *Performance and Tuning: Monitoring and Analyzing*.
- statistics i/o includes the actual total I/O cost of a statement according to the optimizer's costing formula. This value is a number representing the sum of the number of logical I/Os multiplied by the cost of a logical I/O and the number of physical I/Os multiplied by the cost of a physical I/O. For more information on these numbers, see "How Is "Fast" Determined?" in the *Performance and Tuning Guide*.

Defining time ranges

A *time range* is a contiguous block of time within a single day across one or more contiguous days of the week. It is defined by its starting and ending periods.

Adaptive Server includes predefined "at all times" range, which covers the period midnight through midnight, Monday through Sunday. You can create, modify, and drop additional time ranges as necessary for resource limits.

Named time ranges may overlap. However, the limits for a particular user/application combination may not be associated with named time ranges that overlap. You can create different limits that share the same time range.

For example, assume that you limit "joe_user" to returning 100 rows when he is running the payroll application during business hours. Later, you attempt to limit his row retrieval during peak hours, which overlap with business hours. You will get a message that the new limit failed, because it would have overlapped with an existing limit.

Although you cannot limit the row retrieval for "joe_user" in the payroll application during overlapping time ranges, nothing stops you from putting a second limit on "joe_user" during the same time range as the row retrieval limit. For example, you can limit the amount of time one of his queries can run to the same time range that you used to limit his row retrieval.

When you create a named time range, Adaptive Server stores it in the systimeranges system table to control when a resource limit is active. Each time range has a range ID number. The "at all times" range is range ID 1. Adaptive Server messages refer to specific time ranges.

Determining the time ranges you need

Use a chart like the one below to determine the time ranges to create for each server. Monitor server usage throughout the week; then indicate the periods when your server is especially busy or is performing crucial tasks that should not be interrupted.

Day **Time**	**00:00**	**01:00**	**02:00**	**03:00**	**04:00**	**05:00**	**06:00**	**07:00**	**08:00**	**09:00**	**10:00**	**11:00**	**12:00**	**13:00**	**14:00**	**15:00**	**16:00**	**17:00**	**18:00**	**19:00**	**20:00**	**21:00**	**22:00**	**23:00**	**00:00**
Mon																									
Tues																									
Wed																									
Thurs																									
Fri																									
Sat																									
Sun																									

Creating named time ranges

Create new time ranges use sp_add_time_range to:

- Name the time range
- Specify the days of the week to begin and end the time range
- Specify the times of the day to begin and end the time range

For syntax and detailed information, see sp_add_time_range in the *Reference Manual*.

A time range example

Assume that two critical jobs are scheduled to run every week at the following times.

- Job 1 runs from 07:00 to 10:00 on Tuesday and Wednesday.
- Job 2 runs from 08:00 on Saturday to 13:00 on Sunday.

The following table uses “1” to indicate when job 1 runs and “2” to indicate when job 2 runs:

Time / Day	00:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	00:00
Mon																									
Tues								1	1	1	1														
Wed								1	1	1	1														
Thurs																									
Fri																									
Sat									2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Sun	2	2	2	2	2	2	2	2	2	2	2	2	2	2											

Job 1 can be covered by a single time range, tu_wed_7_10:

```
sp_add_time_range tu_wed_7_10, tuesday, wednesday, "7:00", "10:00"
```

Job 2, however, requires two separate time ranges, for Saturday and Sunday:

```
sp_add_time_range saturday_night, saturday, saturday, "08:00", "23:59"
sp_add_time_range sunday_morning, sunday, sunday, "00:00", "13:00"
```

Modifying a named time range

Use sp_modify_time_range to:

- Specify which time range to modify
- Specify the change to the days of the week
- Specify the change to the times of the day

For syntax and detailed information, see sp_modify_time_range in the *Reference Manual*.

For example, to change the end day of the *business_hours* time range to Saturday, retaining the existing start day, start time, and end time, enter:

```
sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL
```

To specify a new end day and end time for the *before_hours* time range, enter:

```
sp_modify_time_range before_hours, NULL, Saturday, NULL, "08:00"
```

Note You cannot modify the "at all times" time range.

Dropping a named time range

Use sp_drop_time_range to drop a user-defined time range

For syntax and detailed information, see sp_drop_time_range in the *Reference Manual*.

For example, to remove the *evenings* time range from the systimeranges system table in the master database, enter:

```
sp_drop_time_range evenings
```

Note You cannot drop the "at all times" time range or any time range for which resource limits are defined.

When do time range changes take effect?

The active time ranges are bound to a login session at the beginning of each query batch. A change in the server’s active time ranges due to a change in actual time has no effect on a session during the processing of a query batch. In other words, if a resource limit restricts query batches during a given time range, but the query batch begins before that time range becomes active, the query batch that is already running is not affected by the resource limit. However, if you run a second query batch during the same login session, that query batch will be affected by the change in time.

Adding, modifying, and deleting time ranges does not affect the active time ranges for the login sessions currently in progress.

If a resource limit has a transaction as its scope, and a change occurs in the server’s active time ranges while a transaction is running, the newly active time range does not affect the transaction currently in progress.

Identifying users and limits

For each resource limit, you must specify the object to which the limit applies.

You can apply a resource limit to any of the following:

- All applications used by a particular login
- All logins that use a particular application
- A specific application used by a particular login

where *application* is defined as a client program running on top of Adaptive Server, accessed through a particular login. To run an application on Adaptive Server, you must specify its name through the CS_APPNAME connection property using cs_config (an Open Client Client-Library application) or the DBSETLAPP function in Open Client DB-Library. To list named applications running on your server, select the program_name column from the master..sysprocesses table.

For more information about the CS_APPNAME connection property, see the *Open Client Client-Library/C Reference Manual*. For more information on the DBSETLAPP function, see the *Open Client DB-Library/C Reference Manual*.

Identifying heavy-usage users

Before you implement resource limits, run sp_reportstats. The output from this procedure will help you identify the users with heavy system usage. For example:

```
                              sp_reportstats
Name    Since          CPU     Percent CPU  I/O     Percent I/O
------  -----------    -----   ------------ -----   -------------
probe   jun 19 1993    0       0%           0       0%
julie   jun 19 1993    10000   24.9962%     5000    24.325%
jason   jun 19 1993    10002   25.0013%     5321    25.8866%
ken     jun 19 1993    10001   24.9987%     5123    24.9234%
kathy   jun 19 1993    10003   25.0038%     5111    24.865%

                              Total CPU   Total I/O
                              ---------   ---------
                              40006       20555
```

The output above indicates that usage is balanced among the users. For more information on chargeback accounting, see "cpu accounting flush interval" on page 152 and "i/o accounting flush interval" on page 163.

Identifying heavy-usage applications

To identify the applications running on your system and the users who are running them, query the sysprocesses system table in the master database.

The following query determines that isql, payroll, perl, and acctng are the only client programs whose names were passed to the Adaptive Server:

```
                              select spid, cpu, physical_io,
                                substring(user_name(uid),1,10) user_name,
                                hostname, program_name, cmd
                              from sysprocesses
spid  cpu    physical_io  user_name hostname program_name cmd
----  ---    -----------  --------- -------- ------------ ------
  17    4          12748  dbo       sabrina  isql         SELECT
 424    5              0  dbo       HOWELL   isql         UPDATE
 526    0            365  joe       scotty   payroll      UPDATE
 568    1           8160  dbo       smokey   perl         SELECT
 595   10              1  dbo       froth    isql         DELETE
 646    1              0  guest     walker   isql         SELECT
 775    4          48723  joe_user  mohindra acctng       SELECT
```

```
(7 rows affected)
```

Because sysprocesses is built dynamically to report current processes, repeated queries produce different results. Repeat this query throughout the day over a period of time to determine which applications are running on your system.

The CPU and physical I/O values are flushed to the syslogins system table periodically where they increment the values shown by sp_reportstats.

After identifying the applications running on your system, use showplan and statistics io to evaluate the resource usage of the queries in the applications.

If you have configured Adaptive Server to enable resource limits, you can use showplan to evaluate resources used prior to execution and statistics io to evaluate resources used during execution. For information on configuring Adaptive Server to enable resource limits, see "Enabling resource limits" on page 237.

In addition to statistics io, statistics time is also useful for evaluating the resources a query consumes. Use statistics time to display the time it takes to execute each step of the query. For more information, see "Diagnostic Tools for Query Optimization" on page 12-6 in the *Performance and Tuning Guide*.

Choosing a limit type

After you determine the users and applications to limit, you have a choice of three different types of resource limits.

Table 6-1 describes the function and scope of each limit type and indicates the tools that help determine whether a particular query might benefit from this type of limit. In some cases, it may be appropriate to create more than one type of limit for a given user and application. For more information on limit types, see "Understanding limit types" on page 246.

Table 6-1: Resource limit types

Limit type	Use for queries that	Measuring resource usage	Scope	Enforced during
io_cost	Require many logical and physical reads.	Use set showplan on before running the query, to display its estimated I/O cost; use set statistics io on to observe the actual I/O cost.	Query	Pre-execution or execution

Limit type	Use for queries that	Measuring resource usage	Scope	Enforced during
row_count	Return large result sets.	Use the *@@rowcount* global variable to help develop appropriate limits for row count.	Query	Execution
elapsed_time	Take a long time to complete, either because of their own complexity or because of external factors such as server load or waiting for a lock.	Use set statistics time on before running the query, to display elapsed time in milliseconds.	Query batch or transaction	Execution
tempdb_space	Use all space in tempdb when creating work or temporary tables.	Number of pages used in tempdb per session.	Query batch or transaction	Execution

The spt_limit_types system table stores information about each limit type.

Determining time of enforcement

Time of enforcement is the phase of query processing during which Adaptive Server applies a given resource limit. Resource limits occur during:

- Pre-execution – Adaptive Server applies resource limits prior to execution, based on the optimizer's I/O cost estimate. This limit prevents execution of potentially expensive queries. I/O cost is the only resource type that can be limited at pre-execution time.

  When evaluating the I/O cost of data manipulation language (DML) statements within the clauses of a conditional statement, Adaptive Server considers each DML statement individually. It evaluates all statements, even though only one clause will actually be executed.

  A pre-execution time resource limit can have only a query limit scope; that is, the values of the resources being limited at compile time are computed and monitored on a query-by-query basis only.

  Adaptive Server does not enforce pre-execution time resource limits statements in a trigger.

- Execution – Adaptive Server applies resource limits at runtime, and is usually used to prevent a query from monopolizing server and operating system resources. Execution time limits may use more resources (additional CPU time as well as I/O) than pre-execution time limits.

Determining the scope of resource limits

The *scope* parameter specifies the duration of a limit in Transact-SQL statements. The possible limit scopes are query, query batch, and transaction:

- Query – Adaptive Server applies resource limits to any single Transact-SQL statement that accesses the server; for example, select, insert, and update. When you issue these statements within a query batch, Adaptive Server evaluates them individually.

  Adaptive Server considers a stored procedure to be a series of DML statements. It evaluates the resource limit of each statement within the stored procedure. If a stored procedure executes another stored procedure, Adaptive Server evaluates each DML statement within the nested stored procedure at the inner nesting level.

  Adaptive Server checks pre-execution time resource limits with a query scope, one nesting level at a time. As Adaptive Server enters each nesting level, it checks the active resource limits against the estimated resource usage of each DML statement prior to executing any of the statements at that nesting level. A resource limit violation occurs if the estimated resource usage of any DML query at that nesting level exceeds the limit value of an active resource limit. Adaptive Server takes the action that is bound to the violated resource limit.

  Adaptive Server checks execution time resource limits with a query scope against the cumulative resource usage of each DML query. A limit violation occurs when the resource usage of a query exceeds the limit value of an active execution time resource limit. Again, Adaptive Server takes the action that is bound to that resource limit.

- Query batch – query batch consists of one or more Transact-SQL statements; for example, in isql, a group of queries becomes a query batch when executed by a single go command terminator.

  The query batch begins at nesting level 0; each call to a stored procedure increments the nesting level by 1 (up to the maximum nesting level). Each return from a stored procedure decrements the nesting level by 1.

  Only execution time resource limits can have a query batch scope.

  Adaptive Server checks execution time resource limits with a query batch scope against the cumulative resource usage of the statements in each query batch. A limit violation occurs when the resource usage of the query batch exceeds the limit value of an active execution time resource limit. Adaptive Server takes the action that is bound to that resource limit.

- Transaction – Adaptive Server applies limits with a transaction scope to all nesting levels during the transaction against the cumulative resource usage for the transaction.

  A limit violation occurs when the resource usage of the transaction exceeds the limit value of an active execution time resource limit. Adaptive Server takes the action that is bound to that resource limit.

  Only execution time resource limits can have a transaction scope.

  Adaptive Server does not recognize nested transactions when applying resource limits. A resource limit on a transaction begins when *@@trancount* is set to 1 and ends when *@@trancount* is set to 0.

Understanding limit types

There are four types of resource limits that allow you to limit resource usage in different ways.

Limiting I/O cost

I/O cost is based on the number of logical and physical accesses ("reads") used during query processing. To determine the most efficient processing plan prior to execution, the Adaptive Server optimizer uses both logical and physical resources to compute an estimated I/O cost.

Adaptive Server uses the result of the optimizer's costing formula as a "unitless" number; that is, a value not necessarily based on a single unit of measurement (such as seconds or milliseconds).

To set resource limits, you must understand how those limits translate into runtime system overhead. For example, you must know the effect that a query with a cost of *x* logical and of *y* physical I/Os has on a production server.

Limiting io_cost can control I/O intensive queries, including queries that return a large result set. However, if you run a simple query that returns all the rows of a large table, and you do not have current statistics on the table's size, the optimizer may not estimate that the query will exceed the io_cost resource limit. To prevent queries from returning large result sets, create a resource limit on row_count.

The tracking of I/O cost limits may be less precise for partitioned tables than for unpartitioned tables when Adaptive Server is configured for parallel query processing. For more information on using resource limits in parallel queries, see the *Performance and Tuning Guide*.

Identifying I/O costs

To develop appropriate limits for I/O cost, determine the number of logical and physical reads required for some typical queries. Use the following set commands:

- set showplan on displays the optimizer's cost estimate. Use this information to set pre-execution time resource limits. A pre-execution time resource limit violation occurs when the optimizer's I/O cost estimate for a query exceeds the limit value. Such limits prevent the execution of potentially expensive queries.
- set statistics io on displays the number of actual logical and physical reads required. Use this information to set execution time resource limits. An execution time resource limit violation occurs when the actual I/O cost for a query exceeds the limit value.

Statistics for actual I/O cost include access costs only for user tables and worktables involved in the query. Adaptive Server may use other tables internally; for example, it accesses sysmessages to print out statistics. Therefore, there may be instances when a query exceeds its actual I/O cost limit, even though the statistics indicate otherwise.

In costing a query, the optimizer assumes that every page needed will require a physical I/O for the first access and will be found in the cache for repeated accesses. Actual I/O costs may differ from the optimizer's estimated costs, for several reasons.

The estimated cost will be higher than the actual cost if some pages are already in the cache or if the statistics are incorrect. The estimated cost may be lower than the actual cost if the optimizer chooses 16K I/O, and some of the pages are in 2K cache pools, which requires many 2K I/Os. Also, if a big join forces the cache to flush its pages back to disk, repeated access may require repeated physical I/Os.

The optimizer's estimates will not be accurate if the distribution or density statistics are out of date or cannot be used.

Calculating the I/O cost of a cursor

The cost estimate for processing a cursor is calculated at declare cursor time for all cursors except execute cursors, which is calculated when the cursor opens.

Pre-execution time resource limits on I/O cost are enforced at open *cursorname* time for all cursor types. The optimizer recalculates the limit value each time the user attempts to open the cursor.

An execution time resource limit applies to the cumulative I/O cost of a cursor from the time the cursor opens to the time it closes. The optimizer recalculates the I/O limit each time a cursor opens.

For a discussion of cursors, see Chapter 18, “Cursors: Accessing Data Row by Row,” in the *Transact-SQL User’s Guide*.

The scope of the *io_cost* limit type

A resource limit that restricts I/O cost applies only to single queries. If you issue several statements in a query batch, Adaptive Server evaluates the I/O usage for each query. For more information, see “Determining the scope of resource limits” on page 245.

Limiting elapsed time

Elapsed time is the number of seconds, in wall-clock time, required to execute a query batch or transaction. Elapsed time is determined by such factors as query complexity, server load, and waiting for locks.

To help develop appropriate limits for elapsed time use information you have gathered with set statistics time You can limit the elapsed time resource only at execution time.

With set statistics time set on, run some typical queries to determine processing time in milliseconds. Convert milliseconds to seconds when you create the resource limit.

Elapsed time resource limits are applied to all SQL statements in the limit’s scope (query batch or transaction), not just to the DML statements. A resource limit violation occurs when the elapsed time for the appropriate scope exceeds the limit value.

Because elapsed time is limited only at execution time, an individual query will continue to run, even if its elapsed time exceeds the limit. If there are multiple statements in a batch, an elapsed time limit takes effect after a statement violates the limit and before the next statement is executed. If there is only one statement in a batch, setting an elapsed time limit has no effect.

Separate elapsed time limits are not applied to nested stored procedures or transactions. In other words, if one transaction is nested within another, the elapsed time limit applies to the outer transaction, which encompasses the elapsed time of the inner transaction. Therefore, if you are counting the wall-clock running time of a transaction, that running time includes all nested transactions.

The scope of the *elapsed_time* limit type

The scope of a resource limit that restricts elapsed time is either a query batch or transaction. You cannot restrict the elapsed time of a single query. For more information, see "Determining the scope of resource limits" on page 245.

Limiting the size of the result set

The row_count limit type limits the number of rows returned to the user. A limit violation occurs when the number of rows returned by a select statement exceeds the limit value.

If the resource limit issues a warning as its action, and a query exceeds the row limit, the full number of rows are returned, followed by a warning that indicates the limit value; for example:

```
Row count exceeded limit of 50.
```

If the resource limit's action aborts the query batch or transaction or kills the session, and a query exceeds the row limit, only the limited number of rows are returned and the query batch, transaction, or session aborts. Adaptive Server displays a message like the following:

```
Row count exceeded limit of 50.
Transaction has been aborted.
```

The row_count limit type applies to all select statements at execution time. You cannot limit an estimated number of rows returned at pre-execution time.

Determining row count limits

Use the *@@rowcount* global variable to help develop appropriate limits for row count. Selecting this variable after running a typical query can tell you how many rows the query returned.

Applying row count limits to a cursor

A row count limit applies to the cumulative number of rows that are returned through a cursor from the time the cursor opens to the time it closes. The optimizer recalculates the row_count limit each time a cursor opens.

The scope of the *row_count* limit type

A resource limit that restricts row count applies only to single queries, not to cumulative rows returned by a query batch or transaction. For more information, see "Determining the scope of resource limits" on page 245.

Setting limits for tempdb space usage

The tempdb_space resource limit restricts the number of pages a tempdb database can have during a single session. If a user exceeds the specified limit, the session can be terminated or the batch or transaction aborted.

For queries executed in parallel, the tempdb_space resource limit is distributed equally among the parallel threads. For example, if the tempdb_space resource limit is set at 1500 pages and a user executes the following with three-way parallelism, each parallel thread can create a maximum of 500 pages in tempdb:

```
select into #temptable from partitioned_table
```

The SA or DBA sets the tempdb_space limit using sp_add_resource_limit, and drops the tempdb_space limit using sp_drop_resource_limit.

Creating a resource limit

Create a new resource limit with sp_add_resource_limit. The syntax is:

sp_add_resource_limit *name*, *appname*, *rangename*, *limittype*, *limit_value*, *enforced*, *action*, *scope*

Use this system procedure's parameters to:

- Specify the name of the user or application to which the resource limit applies.

  You must specify either a *name* or an *appname* or both. If you specify a user, the name must exist in the syslogins table. Specify "null" to create a limit that applies to all users or all applications.

- Specify the type of limit (io_cost, row_count, elapsed_time, or tempdb_space), and set an appropriate value for the limit type.

  For more information, see "Choosing a limit type" on page 243.

- Specify whether the resource limit is enforced prior to or during query execution.

  Specify numeric values for this parameter. Pre-execution time resource limits, which are specified as 1, are valid only for the io_cost limit. Execution time resource limits, which are specified as 2, are valid for all three limit types. For more information, see "Determining time of enforcement" on page 244.

- Specify the action to be taken (issue a warning, abort the query batch, abort the transaction, or kill the session).

  Specify numeric values for this parameter.

- Specify the scope (query, query batch, or transaction).

  Specify numeric values for this parameter. For more information, see "Determining the scope of resource limits" on page 245.

For detailed information, see sp_add_resource_limit in the *Reference Manual*.

Resource limit examples

This section includes three examples of setting resource limits.

Examples

Example 1 This example creates a resource limit that applies to all users of the payroll application because the name parameter is NULL:

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 120, 2, 1, 2
```

The limit is valid during the tu_wed_7_10 time range. The limit type, elapsed_time, is set to a value of 120 seconds. Because elapsed_time is enforced only at execution time, the *enforced* parameter is set to 2. The *action* parameter is set to 1, which issues a warning. The limit's *scope* is set to 2, query batch, by the last parameter. Therefore, when the elapsed time of the query batch takes more than 120 seconds to execute, Adaptive Server issues a warning.

Example 2 This example creates a resource limit that applies to all ad hoc queries and applications run by "joe_user" during the saturday_night time range:

```
sp_add_resource_limit joe_user, NULL, saturday_night,
row_count, 5000, 2, 3, 1
```

If a query (*scope* = 1) returns more than 5000 rows, Adaptive Server aborts the transaction (*action* = 3). This resource limit is enforced at execution time (*enforced* = 2).

Example 3 This example also creates a resource limit that applies to all ad hoc queries and applications run by "joe_user":

```
sp_add_resource_limit joe_user, NULL, "at all times",
io_cost, 650, 1, 3, 1
```

However, this resource limit specifies the default time range, "at all times." When the optimizer estimates that the io_cost of the query (*scope* = 1) would exceed the specified value of 650, Adaptive Server aborts the transaction (*action* = 3). This resource limit is enforced at pre-execution time (*enforced* = 1).

Note Although Adaptive Server terminates the current transaction when it reaches its time limit, you receive no 1105 error message until you issue another SQL command or batch; in other words, the message appears only when you attempt to use the connection again.

Getting information on existing limits

Use sp_help_resource_limit to get information about existing resource limits.

Users who do not have the System Administrator role can use sp_help_resource_limit to list their own resource limits (only).

Users either specify their own login names as a parameter or specify the *name* parameter as “null.” The following examples return all resource limits for user “joe_user” when executed by joe_user:

```
sp_help_resource_limit
```

or

```
sp_help_resource_limit joe_user
```

System Administrators can use sp_help_resource_limit to get the following information:

- All limits as stored in sysresourcelimits (all parameters NULL); for example:

  ```
  sp_help_resource_limit
  ```

- All limits for a given login (*name* is not NULL, all other parameters are NULL); for example:

  ```
  sp_help_resource_limit joe_user
  ```

- All limits for a given application (*appname* is not NULL; all other parameters are NULL); for example:

  ```
  sp_help_resource_limit NULL, payroll
  ```

- All limits in effect at a given time or day (either *limittime* or *limitday* is not NULL; all other parameters NULL); for example:

  ```
  sp_help_resource_limit @limitday = wednesday
  ```

- Limit, if any, in effect at a given time for a given login (*name* is not NULL, either *limittime* or *limitday* is not NULL); for example:

  ```
  sp_help_resource_limit joe_user, NULL, NULL,
  wednesday
  ```

For detailed information, see sp_help_resource_limit in the *Reference Manual*.

Example of listing all existing resource limits

When you use sp_help_resource_limit without any parameters, Adaptive Server lists all resource limits within the server. For example:

```
                sp_help_resource_limit
name     appname rangename rangeid limitid limitvalue enforced  action scope
----     ------- --------- ------- ------- ---------- --------  ------ -----
NULL     acctng  evenings        4       2        120        2       1     2
```

```
stein    NULL    weekends        1       3      5000          2       1     1
joe_user acctng  bus_hours       5       3      2500          2       2     1
joe_user finance bus_hours       5       2       160          2       1     6
wong     NULL    mornings        2       3      2000          2       1     1
wong     acctng  bus_hours       5       1        75          1       3     1
```

In the output, the rangeid column prints the value from systimeranges.id that corresponds to the name in the rangename column. The limitvalue column reports the value set by sp_add_resource_limit or sp_modify_resource_limit. Table 6-2 shows the meaning of the values in the limitid, enforced, action, and scope columns.

Table 6-2: Values for sp_help_resource_limit output

Column	Meaning	Value
limitid	What kind of limit is it?	1– I/O cost 2 – Elapsed time 3 – Row count
enforced	When is the limit enforced?	1 – Before execution 2 – During execution 3 – Both
action	What action is taken when the limit is hit?	1– Issue a warning 2 – Abort the query batch 3 – Abort the transaction 4 – Kill the session
scope	What is the scope of the limit?	1 – Query 2 – Query batch 4 – Transaction 6 – Query batch + transaction

If a System Administrator specifies a login name when executing sp_help_resource_limit, Adaptive Server lists all resource limits for that login. The output displays not only resource limits specific to the named user, but all resource limits that pertain to all users of specified applications, because the named user is included among all users.

For example, the following output shows all resource limits that apply to “joe_user”. Because a resource limit is defined for all users of the acctng application, this limit is included in the output.

```
                           sp_help_resource_limit joe_user
name     appname rangename rangeid limitid limitvalue enforced  action scope
----     ------- --------- ------- ------- ---------- --------  ------ -----
```

```
NULL     acctng  evenings        4       2       120          2       1     2
joe_user acctng  bus_hours       5       3      2500          2       2     1
joe_user finance bus_hours       5       2       160          2       1     6
```

Modifying resource limits

Use sp_modify_resource_limit to specify a new limit value or a new action to take when the limit is exceeded or both. You cannot change the login or application to which a limit applies or specify a new time range, limit type, enforcement time, or scope.

The syntax of sp_modify_resource_limit is:

> sp_modify_resource_limit *name*, *appname*, *rangename*, *limittype*, *limitvalue*, *enforced*, *action*, *scope*

To modify a resource limit, specify the following values:

- You must specify a non-null value for either *name* or *appname*.
    - To modify a limit that applies to all users of a particular application, specify a *name* of "null."
    - To modify a limit that applies to all applications used by *name*, specify an *appname* of "null."
    - To modify a limit that governs a particular application, specify the application name that the client program passes to the Adaptive Server in the login packet.
- You must specify non-null values for *rangename* and *limittype*. If necessary to uniquely identify the limit, specify non-null values for *action* and *scope*.
- Specifying "null" for *limitvalue* or *action* indicates that its value does not change.

For detailed information, see sp_modify_resource_limit in the *Reference Manual*.

Examples of modifying a resource limit

```
sp_modify_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 90, null, null, 2
```

This example changes the value of the resource limit that restricts elapsed time to all users of the *payroll* application during the tu_wed_7_10 time range. The limit value for elapsed time decreases to 90 seconds (from 120 seconds). The values for time of execution, action taken, and scope remain unchanged.

```
sp_modify_resource_limit joe_user, NULL,
saturday_night, row_count, NULL, NULL, 2, NULL
```

This example changes the action taken by the resource limit that restricts the row count of all ad hoc queries and applications run by “joe_user” during the saturday_night time range. The previous value for action was 3, which aborts the transaction when a query exceeds the specified row count. The new value is to 2, which aborts the query batch. The values for limit type, time of execution, and scope remain unchanged.

Dropping resource limits

Use sp_drop_resource_limit to drop a resource limit from an Adaptive Server.

The syntax is:

sp_drop_resource_limit {*name* , *appname* } [, *rangename*, *limittype*, *enforced*, *action*, *scope*]

Specify enough information to uniquely identify the limit. You must specify a non-null value for either *name* or *appname*. In addition, specify values according to those shown in Table 6-3.

Table 6-3: Identifying resource limits to drop

Parameter	Value specified	Consequence
name	• Specified login	Drops limits that apply to the particular login.
	• NULL	Drops limits that apply to all users of a particular application.
appname	• Specified application	Drops limits that apply to a particular application.
	• NULL	Drops limits that apply to all applications used by the specified login.
timerange	• An existing time range stored in the systimeranges system table	Drops limits that apply to a particular time range.
	• NULL	Drops all resource limits for the specified *name, appname, limittype,* enforcement time, *action,* and *scope,* without regard to *rangename*.

Parameter	Value specified	Consequence
limittype	• One of the three limit types: row_count, elapsed_time, io_cost	Drops limits that apply to a particular limit type.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *action*, and *scope*, without regard to *limittype*.
enforced	• One of the enforcement times: pre-execution or execution	Drops the limits that apply to the specified enforcement time.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *limittype*, *timerange*, *action*, and *scope*, without regard to enforcement time.
action	• One of the four action types: issue warning, abort query batch, abort transaction, kill session	Drops the limits that apply to a particular action type.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *limittype*, enforcement time, and *scope*, without regard to *action*.
scope	• One of the scope types: query, query batch, transaction	Drops the limits that apply to a particular scope.
	• NULL	Drops all resource limits for the specified *name*, *appname*, *timerange*, *limittype*, enforcement time, and *action*, without regard to *scope*.

When you use sp_droplogin to drop an Adaptive Server login, all resource limits associated with that login are also dropped.

For detailed information, see sp_drop_resource_limit in the *Reference Manual*.

Examples of dropping a resource limit

Example 1 Drops all resource limits for all users of the payroll application during the tu_wed_7_10 time range:

```
sp_drop_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time
```

Example 2 Is similar to the preceding example, but drops only the resource limit that governs elapsed time for all users of the payroll application during the tu_wed_7_10 time range:

```
sp_drop_resource_limit NULL, payroll, tu_wed_7_10
```

Example 3 Drops all resource limits for “joe_user” from the payroll application:

```
sp_drop_resource_limit joe_user, payroll
```

Resource limit precedence

Adaptive Server provides precedence rules for time ranges and resource limits.

Time ranges

For each login session during the currently active time ranges, only one limit can be active for each distinct combination of limit type, enforcement time, and scope. The precedence rules for determining the active limit are as follows:

- If no limit is defined for the login ID for either the "at all times" range or the currently active time ranges, there is no active limit.
- If limits are defined for the login for both the "at all times" and time-specific ranges, then the limit for the time-specific range takes precedence.

Resource limits

Since either the user's login name or the application name, or both, are used to identify a resource limit, Adaptive Server observes a predefined search precedence while scanning the sysresourcelimits table for applicable limits for a login session. The following table describes the precedence of matching ordered pairs of login name and application name:

Level	Login name	Application name
1	joe_user	payroll
2	NULL	payroll
3	joe_user	NULL

If one or more matches are found for a given precedence level, no further levels are searched. This prevents conflicts regarding similar limits for different login/application combinations.

If no match is found at any level, no limit is imposed on the session.

CHAPTER 7

Configuring Character Sets, Sort Orders, and Languages

This chapter discusses Adaptive Server Enterprise internationalization and localization support issues.

Understanding internationalization and localization

Internationalization is the process of enabling an application to support multiple languages and cultural conventions.

An internationalized application uses external files to provide language-specific information at execution time. Because it contains no language-specific code, an internationalized application can be deployed in any native language environment without code changes. A single version of a software product can be adapted to different languages or regions, conforming to local requirements and customs without engineering changes. This approach to software development saves significant time and money over the lifetime of an application.

Localization is the process of adapting an internationalized product to meet the requirements of one particular language or region, for example Spanish, including providing translated system messages; translations for the user interface; and the correct formats for date, time, and currency. One version of a software product may have many localized versions.

Sybase provides both internationalization and localization support. Adaptive Server includes the character set definition files and sort order definition files required for data processing support for the major business languages in Western Europe, Eastern Europe, the Middle East, Latin America, and Asia.

Sybase Language Modules provide translated system messages and formats for Chinese (Simplified), French, German, Japanese, Korean, Brazilian Portuguese, and Spanish. By default, Adaptive Server comes with U.S. English message files.

This chapter describes the available character sets and language modules and summarizes the steps needed to change the default character set, sort order, or message language for Adaptive Server.

Advantages of internationalized systems

The task of designing an application to work outside its country of origin can seem daunting. Often, programmers think that internationalizing means hard-coding dependencies based on cultural and linguistic conventions for just one country.

A better approach is to write an internationalized application: that is, one that examines the local computing environment to determine what language to use and loads files containing language-specific information at runtime.

When you use an internationalized application, a single application can be deployed in all countries. This has several advantages:

- You write and maintain one application, not half a dozen (or more).
- The application can be deployed, without change, in new countries as needed. You need only supply the correct localization files.
- All sites can expect standard features and behavior.

A sample internationalized system

An internationalized system may include internationalized client applications, gateways, and servers running on different platforms in different native language environments.

For example, an international system might include the following components:

- Order processing applications in New York City, Mexico City, and Paris (Client-Library applications)
- An inventory control server in Germany (Adaptive Server)
- An order fulfillment server in France (Adaptive Server)
- A central accounting application in Japan (an Open Server application working with an Adaptive Server)

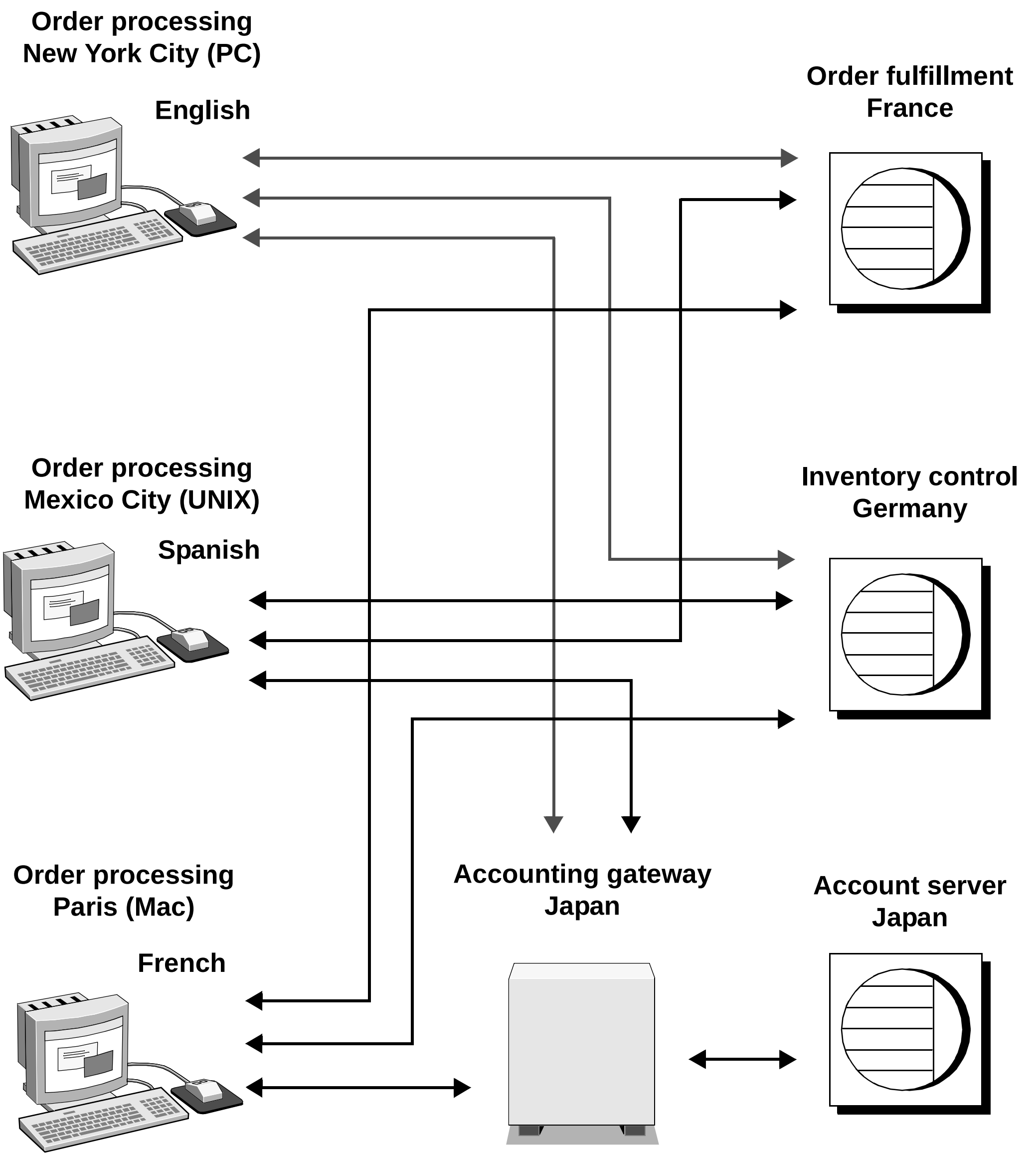
In this system, the order processing applications:

- Query the inventory control server to determine if requested items are in stock
- Place orders with the order fulfillment server
- Send financial information to the accounting application

The inventory control server and the order fulfillment server respond to queries, and the accounting application collects financial data and generates reports.

The system looks like this:

Figure 7-1: Example of an international system



In this example, all applications and servers use local languages and character sets to accept input and output messages.

Elements of an internationalized system

There are three elements that you can manipulate to configure your server language in an internationalized environment. Sybase suggests that you review these three elements and carefully plan the client/server network you want to create.

- Character set – the language in which the server sends and receives data to and from the client servers. Select the character set after carefully planning and analyzing the language needs of all client servers.
- Sort order – sort order options are dependent on the language and character set you select.
- System messages – messages display in one of several languages provided by Sybase. If your server language is not one of the languages provided, your system messages display in English, the default.

The following sections provide details about each of these elements.

Selecting the character set for your server

All data is encoded in your server in a special code. For example, the letter “a” is encoded as “97” in decimal. A **character set** is a specific collection of characters (including alphabetic and numeric characters, symbols, and nonprinting control characters) and their assigned numerical values, or codes. A character set generally contains the characters for an alphabet, for example, the Latin alphabet used in the English language, or a script such as Cyrillic used with languages such as Russian, Serbian, and Bulgarian. Character sets that are platform-specific and support a subset of languages, for example, the Western European languages, are called **native** or **national character sets**. All character sets that come with Adaptive Server, except for Unicode UTF-8, are native character sets.

A **script** is a writing system, a collection of all the elements that characterize the written form of a human language—for example, Latin, Japanese, or Arabic. Depending on the languages supported by an alphabet or script, a character set can support one or more languages. For example, the Latin alphabet supports the languages of Western Europe (see Group 1 in Table 7-1 on page 264). On the other hand, the Japanese script supports only one language, Japanese. Therefore, the Group 1 character sets support multiple languages, while many character sets, such as those in Group 101, support only one language.

The language or languages that are covered by a character set is called a **language group.** A language group can contain many languages or only one language; a native character set is the platform-specific encoding of the characters for the language or languages of a particular language group.

Within a client/server network, you can support data processing in multiple languages *if all the languages belong to the same language group* (see Table 7-1 on page 264). For example, if data in the server is encoded in a Group 1 character set, you could have French, German, and Italian data and any of the other Group 1 languages in the same database. However, you cannot store data from another language group in the same database. For example, you cannot store Japanese data with French or German data.

Unlike the native character sets just described, **Unicode** is an international character set that supports over 650 of the world's languages, such as Japanese, Chinese, Russian, French, and German. Unicode allows you to mix different languages from different language groups in the same server, no matter what the platform. See "Unicode" on page 266 for more information.

Since all character sets support the Latin script, and therefore English, a character set always supports at least two languages—English and one other language.

Many languages are supported by more than one character set. The character set you install for a language depends on the client's platform and operating system.

Adaptive Server supports the following languages and character sets:

Table 7-1: Supported languages and character sets

Language group	Languages	Character sets
Group 1	**Western European:** Albanian, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish, Swedish	ASCII 8, CP 437, CP 850, CP 860, CP 863, CP 1252[a] , ISO 8859-1, ISO 8859-15, Macintosh Roman, ROMAN8

Language group	Languages	Character sets
Group 2	**Eastern European:** Croatian, Czech, Estonian, Hungarian, Latvian, Lithuanian, Polish, Romanian, Slovak, Slovene (and English)	CP 852, CP 1250, ISO 8859-2, Macintosh Central European
Group 4	Baltic (and English)	CP 1257
Group 5	**Cyrillic:** Bulgarian, Byelorussian, Macedonian, Russian, Serbian, Ukrainian (and English)	CP 855, CP 866, CP 1251, ISO 8859-5, Koi8, Macintosh Cyrillic
Group 6	Arabic (and English)	CP 864, CP 1256, ISO 8859-6
Group 7	Greek (and English)	CP 869, CP 1253, GREEK8, ISO 8859-7, Macintosh Greek
Group 8	Hebrew (and English)	CP 1255, ISO 8859-8
Group 9	Turkish (and English)	CP 857, CP 1254, ISO 8859-9, Macintosh Turkish, TURKISH8
Group 101	Japanese (and English)	CP 932 DEC Kanji, EUC-JIS, Shift-JIS
Group 102	Simplified Chinese (PRC) (and English)	CP 936, EUC-GB
Group 103	Traditional Chinese (ROC) (and English)	Big 5, CP 950[b] , EUC-CNS
Group 104	Korean (and English)	EUC-KSC
Group 105	Thai (and English)	CP 874, TIS 620
Group 106	Vietnamese (and English)	CP 1258
Unicode	Over 650 languages	UTF-8

a. CP 1252 is identical to ISO 8859-1 except for the 0x80–0x9F code points which are mapped to characters in CP 1252.
b. CP 950 is identical to Big 5.

Note The English language is supported by all character sets because the first 128 (decimal) characters of any character set include the Latin alphabet (defined as “ASCll-7”). The characters beyond the first 128 differ between character sets and are used to support the characters in different native languages. For example, code points 0-127 of CP 932 and CP 874 both support English and the Latin alphabet. However, code points 128-255 support Japanese characters in CP 932 and code points 128-255 support Thai characters in CP 874.

The following character sets support the European currency symbol, the “euro”: CP 1252 (Western Europe); CP 1250 (Eastern Europe); CP 1251 (Cyrillic); CP 1256 (Arabic); CP 1253 (Greek); CP 1255 (Hebrew); CP 1254 (Turkish); CP 874 (Thai); and Unicode UTF-8.

To mix languages from different language groups you *must* use Unicode. If your server character set is Unicode, you can support more than 650 languages in a single server and mix languages from any language group.

Unicode

Unicode is the first character set that enables all the world's languages to be encoded in the same data set. Prior to the introduction of Unicode, if you wanted to store data in, for example, Chinese, you had to choose a character set appropriate for that language—to the exclusion of most other languages. It was either impossible or impractical to mix character sets, and thus diverse languages, in the same data set.

In Adaptive Server version 12.5, Sybase supported Unicode in the form of two new datatypes: unichar and univarchar. These datatypes store data in the UTF-16 encoding of Unicode.

UTF-16 is an encoding wherein Unicode scalar values are represented by a single 16-bit value (or, in rare cases, as a pair of 16-bit values). The two encodings are equivalent insofar as either encoding can be used to represent any Unicode character. The choice of UTF-16 datatypes, rather than a UTF-16 server default character set, was intended to promote easy, step-wise migration for existing database applications.

Adaptive Server version 12.5.1 supports Unicode literals in SQL queries and a wide range of sort orders for UTF-8.

The character set model used by Adaptive Server is based on a single, configurable, server-wide character set. All data stored in Adaptive Server, using any of the "character" datatypes (char, varchar, nchar, nvarchar, and text), is interpreted as being in this character set. Sort orders are defined using this character set, as are language modules—collections of server messages translated into local languages.

During the connection dialog, a client application declares its native character set and language. If properly configured, the server thereafter attempts to convert any character data between its own character set and that of the client (character data includes any data stored in the database, as well as server messages in the client's native language).This works well as long as the server's and client's character sets are compatible. It does not work well when characters are not defined in the other character set, as would be the case for the character sets SJIS, used for Japanese, and KOI8, used for Russian and other Cyrillic languages. Such incompatibilities are the reason for Unicode, which can be thought of as a character superset, including definitions for characters in all other character sets.

In Adaptive Server version 12.5, the character set model was not modified. Rather, a separate mechanism was added whereby data could be stored and manipulated in Unicode. The new Unicode datatypes unichar and univarchar are completely independent of the traditional character set model. Clients send and receive Unicode data independently of whatever other character data they send and receive.

Character set installation

As of Adaptive Server version 12.5.1, the 4-byte form of UTF-8 is supported. This form is used to represent the same rare Unicode characters that are represented in UTF-16 by pairs of 16-bit values ("surrogate pairs"). Prior to Adaptive Server version 12.5.1, only the 3-byte forms of UTF-8 were supported. If you have installed the UTF-8 character set in an Adaptive Server server earlier than version 12.5.1, you should reinstall it to enable the use of the 4-byte form of UTF-8.

Configuration parameters

The UTF-16 encoding of Unicode includes "surrogate pairs," which are pairs of 16-bit values that represent infrequently used characters. Additional checking is built in to Adaptive Server to ensure the integrity of surrogate pairs. You can switch this checking off by setting the configuration parameter "enable surrogate processing" to 0. This yields slightly higher performance, although the integrity of surrogate pairs is no longer guaranteed.

Unicode also defines "normalization," which is the process by which all possible representations of a single character are transformed into a single representation. Many base characters followed by combining diacritical marks are equivalent to precomposed characters, although their bit patterns are different. For example, the following two sequences are equivalent:

```
0x00E9  -- é (LATIN SMALL LETTER E WITH ACUTE)

0x00650301  -- e (LATIN SMALL LETTER E), ´ (COMBINING ACUTE ACCENT)
```

The enable unicode normalization configuration parameter controls whether or not Adaptive Server normalizes incoming Unicode data.

Significant performance increases are possible when the default Unicode sortorder is set to "binary" and the enable Unicode normalization configuration parameter is set to 1. This combination allows Adaptive Server to make several assumptions about the nature of the Unicode data, and code has been implemented to take advantage of these assumptions.

Functions

All built-in functions taking char parameters have been overloaded to accept unichar as well. Built-in functions with more than one parameter, when called with at least one unichar parameter, results in implicit conversion of any non-unichar parameters to unichar.

To guarantee the integrity of surrogate pairs when enable surrogate processing is set to 1 (the default), the string functions do not allow surrogate pairs to be split. Positions are modified to fall at the beginning of a surrogate pair.

Several new built-in functions have been added to round out the unichar support. Included are the functions to_unichar() and uscalar(), which are analogous to char() and ascii(). The functions uhighsurr() and ulowsurr() allow the explicit handling of surrogate pairs in user code.

Using unichar columns

When using the isql or bcp utilities, Unicode values display in hexadecimal form unless the -Jutf8 flag is used, indicating the client's character set is UTF-8. In this case, the utility converts any Unicode data it receives from the server into UTF-8. For example:

```
% isql -Usa -P -Jiso_1
1> select unicode_name from people where unicode_name = 'Jones'
2> go

unicode_name
------------------------------------------------------------------|
0x004a006f006e00650073
(1 row affected)
```

whereas:

```
% isql -Usa -P -Jutf8
1> select unicode_name from people where unicode_name = 'Jones'
2> go

unicode_name
------------------------------------------------------------------
Jones
(1 row affected)
```

This facilitates ad hoc queries. Not all terminal windows are capable of displaying the full repertoire of Unicode characters, but simple tests involving ASCII characters are greatly simplified.

Open Client interoperability

The Open Client libraries have been enhanced to support the new datatype cs_unichar, which can be bound to user variables declared as an array of short integers. This Open Client datatype interfaces directly with the server's Unicode 's unichar and univarchar.

Java interoperability

The internal JDBC driver has been modified to efficiently transfer unichar data between SQL and Java contexts.

Going from SQL to Java, the class java.sql.ResultSet provides a number of “get” methods to retrieve data from the columns of a result set. Any of these get methods work with columns defined as unichar or univarchar. The method getString() is particularly efficient since no conversion needs to be performed.

Going from Java to SQL should be done using the setString() method of the class java.sql.PreparedStatement. The internal JDBC driver copies Java string data directly into the SQL parameter defined as unichar or univarchar.

The external JDBC driver (jConnect) has been modified to support the same seamless interface as the internal driver.

Limitations

Due to the lack of a Unicode-based language parser in Adaptive Server 12.5, a restriction was imposed on the use of the new Unicode datatypes. To use the new datatypes, the server required its default character set to be configured as UTF-8. This restriction has been removed in Adaptive Server version 12.5.1. Unicode datatypes can be used regardless of the server’s default character set.

There is presently no implementation of the datatype unitext, the analog of the text datatype. With the relaxation of the stringent 255-byte column width (a feature of Adaptive Server 12.5), the need for a large Unicode datatype is somewhat reduced. The unitext datatype may be implemented in a future version of Adaptive Server. In the meantime, a temporary workaround is possible. With suitable care, the image datatype may be used to store large blocks of Unicode data.

Selecting the server default character set

When you configure your server, you are asked to specify a default character set for the server. The default character set is the character set in which the server stores and manipulates data. Each server can have only one default character set.

By default, the installation tool assumes that the native character set of the platform operating system is the server's default character set. However, you can select any character set supported by Adaptive Server as the default on your server (see Table 7-1 on page 264).

For example, if you are installing the server on IBM RS/6000 running AIX, and you select one of the Western European languages to install, the installation tool assumes the default character set to be ISO 8859-1.

If you are installing a Unicode server, select UTF–8 as your default character set.

For non-Unicode servers, determine what platform most of your client systems use and use the character set for this platform as the default character set on the server.

This has two advantages:

- The number of unmappable characters between character sets is minimized.

  Since there is usually not a complete one-to-one mapping between the characters in two character sets, there is a potential for some data loss. This is usually minor because most nonconverted characters are special symbols that are not commonly used or are specific to a platform.

- This minimizes the character set conversion that is required.

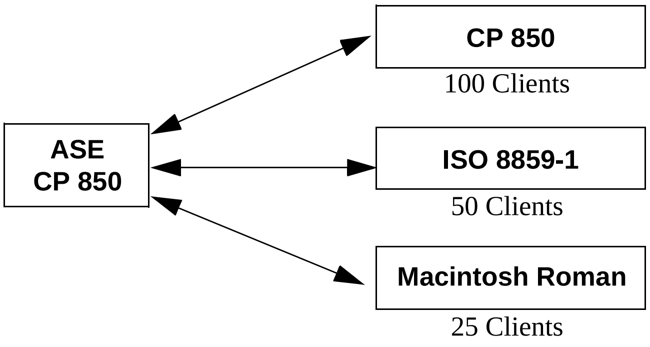
  When the character set on the client system differs from the default character set on the server, data must be converted in order to ensure data integrity. Although the measured performance decrease that results from character set conversion is insignificant, it is good practice to select the default character set that results in the fewest conversions.

For example, if most of your clients use CP850, specify CP850 on your server. You can do this even if your server is on an HP-UX system (where its native character set for the Group 1 languages is ROMAN8).

Note Sybase strongly recommends that you decide which character set you want to use as your default before you create any databases or make any changes to the Sybase-supplied databases.

In the example below in Figure 7-2, 175 clients all access the same Adaptive Server. The clients are on different platforms and use different character sets. The critical factor that allows these clients to function together is that *all* of the character sets in the client/server system belong to the same language group (see Table 7-1 on page 264). Notice that the default language for the Adaptive Server is CP 850, which is the character set used by the largest number of clients. This allows the server to operate most efficiently, with the least amount of character set conversion.

Figure 7-2: Clients using different character sets in the same language group



To help you choose the default character set for your server, the following tables list the most commonly used character sets by platform and language.

Table 7-2: Popular Western European client platforms

Platform	Language	Character set
Win 95, 98	U.S. English, Western Europe	CP 1252
Win NT 4.0	U.S. English, Western Europe	CP 1252
Win 2000	U.S. English, Western Europe	CP 1252
Sun Solaris	U.S. English, Western Europe	ISO 8859-1
HP-UX 10,11	U.S. English, Western Europe	ISO 8859-1
IBM AIX 4.x	U.S. English, Western Europe	ISO 8859-1

Table 7-3: Popular Japanese client platforms

Platform	Language	Character set
Win 95, 98	Japanese	CP 932 for Windows
Win NT 4.0	Japanese	CP 932 for Windows
Win 2000	Japanese	CP 932 for Windows
Sun Solaris	Japanese	EUC-JIS
HP-UX 10,11	Japanese	EUC-JIS
IBM AIX 4.x	Japanese	EUC-JIS

Table 7-4: Popular Chinese client platforms

Platform	Language	Character set
Win 95, 98	Chinese (simplified)	CP 936 for Windows
Win NT 4.0	Chinese (simplified)	CP 936 for Windows
Win 2000	Chinese (simplified)	CP 936 for Windows
Sun Solaris	Chinese (simplified)	EUC-GB
HP-UX 10,11	Chinese (simplified)	EUC-GBS
IBM AIX 4.x	Chinese (simplified)	EUC-GB

Selecting the sort order

Different languages sort the same characters differently. For example, in English, *Cho* would be sorted before *Co*, whereas in Spanish, the opposite is true. In German, β is a single character, however in dictionaries it is treated as the double character *ss* and sorted accordingly. Accented characters are sorted in a particular order so that *aménité* comes before *amène*, whereas if you ignored the accents, the reverse would be true. Therefore, language-specific sort orders are required so that characters are sorted correctly.

Each character set comes with one or more sort orders that Adaptive Server uses to collate data. A sort order is tied to a particular language or set of languages and to a specific character set. The same sort orders can be used for English, French, and German because they sort the same characters identically, for example, *A*, *a*, *B*, *b*, and so on. Or the characters are specific to one of the languages—for example, the accented characters, *é* , *à*, and *á*, are used in French but not in English or German—and therefore, there is no conflict in how those characters are sorted. The same is not true for Spanish however, where the double letters *ch* and *ll* are sorted differently. Therefore, although the same character sets support all four languages, there is one set of sort orders for English, French and German, and a different set of sort orders for Spanish.

In addition, a sort order is tied to a particular character set. Therefore, there is one set of sort orders for English, French, and German in the ISO 8859-1 character set, another set in the CP 850 character set, and so on. The sort orders available for a particular character set are located in sort order definition files (**.srt* files) in the character set directory. For a list of character sets and their available sort orders, see Table 7-5 on page 276.

Using sort orders

Sort orders are used to:

- Create Indexes
- Store data into indexed tables
- Specify an order by clause

Different types of sort orders

All character sets are offered with a binary sort order at a minimum, which blindly sorts all data based only on the arithmetic value of the code assigned to represent each letter (the “binary” code) in the character set. Binary sort order works well for the first 128 characters of each character set (ASCII English) and for Asian languages.When a character set supports more than one language (for example, Group 1 or Unicode) the binary sort order will most likely give incorrect results, and you should select another sort order.

Character sets may also have one or more of the following dictionary sort orders:

- *Dictionary order, case-sensitive, accent-sensitive* – sorts uppercase and lowercase letters separately. Dictionary order recognizes the various accented forms of a letter and sorts them after the associated unaccented letter.
- *Dictionary order, case-insensitive, accent-sensitive* – sorts data in dictionary order but does not recognize case differences. Uppercase letters are equivalent to their lowercase counterparts and are intermingled in sorting results. Useful for avoiding duplicate entries in tables of names.
- *Dictionary order, case-insensitive, accent-sensitive, order with preference* – does not recognize case difference in determining equivalency of items. A word in uppercase is equivalent to the same word in lowercase. Preference is given to uppercase letters (they appear first) if all other conditions are equal.

  Using case-insensitive with preference may cause poor performance in large tables when the columns specified in an *order by* clause match the key of the table's clustered index. Do not select case-insensitive order with preference unless your installation requires that uppercase letters be sorted before lowercase letters in otherwise equivalent strings for *order by* clauses.
- *Dictionary order, case-insensitive, accent-insensitive* – treats accented forms of a letter as equivalent to the associated unaccented letter. It intermingles accented letters in sorting results.

Selecting the default sort order

Sybase servers can support only one default sort order at a time. If your users are using the same language or their languages use the same sort order, then select the desired sort order. For example, if your users are using French data and expect French sorting, then you can pick one of the French dictionary sort orders. Or if your users are using data in multiple languages and the languages use the same sort order, for example English, French, and German, you can pick one sort order and it will work for all your users in all languages.

However, if you have users using different languages that require different sort orders, for example French and Spanish, then you must select one of the sort orders as the default. If you pick, for example, a French sort order, your Spanish users will not see the *ch* and *ll* double characters sorted as they would expect. The installation procedure, by default, configures the server with the binary sort order.

You can use the sortkey function to setup customized alternative sort orders for your data—one for each language.These sort orders can be selected dynamically to meet the needs of different users. The sortkey function is separate from the default sort order, but can coexist in the same server. The range and depth of sort orders provided by the sortkey function is better than those provided by the default sort order mechanism. For more information, see sortkey and compare in the *Reference Manual*.

Table 7-5: Available sort orders

Language or script	Character sets	Sort orders
All languages	UTF-8	Binary
Cyrillic: Bulgarian, Byelorussian, Macedonian, Russian, Serbian, Ukrainian	CP 855, CP 866, CP 1251, ISO 8859-5, Koi8, Macintosh Cyrillic	Dictionary order, case sensitive, accent sensitive
English, French, German	ASCII 8, CP 437, CP850, CP 860, CP 863, CP 1252a, ISO 8859-1, ISO 8859-15, Macintosh Roman, ROMAN8	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, accent sensitive Dictionary order, case sensitive, accent sensitive, with preference Dictionary order, case insensitive, accent insensitive
English, French, German	CP 850	Alternate dictionary order, case sensitive Alternate dictionary order, case sensitive, accent insensitive Alternate dictionary order, case sensitive, with preference
Greek	ISO 8859-7	Dictionary order, case sensitive, accent sensitive
Hungarian	ISO 8859-2	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, accent sensitive Dictionary order, case insensitive, accent insensitive
Russian	CP 866, CP 1251, ISO 8859-5, Koi8, Macintosh Cyrillic	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, accent sensitive
Scandinavian	CP 850	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, with preference
Spanish	ASCII 8, CP 437, CP850, CP 860, CP 863, CP 1252, ISO 8859-1, ISO 8859-15, Macintosh Roman, ROMAN8	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, accent sensitive Dictionary order, case insensitive, accent insensitive
Thai	CP 874, TIS 620	Dictionary order
Turkish	ISO 8859-9	Dictionary order, case sensitive, accent sensitive Dictionary order, case insensitive, accent insensitive Dictionary order, case insensitive, accent sensitive

If your language does not appear here, there is no language-specific sort order for your language. Select a binary sort order and then investigate whether the sortkey function meets your needs. As this table illustrates, many languages have more than one sort order.

Selecting the default Unicode sort order

The default Unicode sort order is distinctly different from the sort order for the server's default character set. This separate configuration parameter is a static parameter that requires that you restart your server and reindex the unichar data if it is changed. This sort order is identified using a string parameter, rather than a numeric parameter, to guarantee that the sort order is unique.

Table 7-6 lists the available default Unicode sort orders.

Table 7-6: Default Unicode sort orders

Name	ID	Description
defaulml	20	Default Unicode multilingual ordering
thaidict	21	Thai dictionary ordering
iso14651	22	ISO 14651 ordering
utf8bin	24	Intended only for unichar (UTF-16); matches Unicode UTF-8 binary ordering
binary	25	UTF-16 binary ordering
altnoacc	39	Alternate case- and accent-insensitive dictionary ordering
altdict	45	Alternate dictionary ordering
altnocsp	46	Alternate dictionary sorting with case insensitivity and preference
scandict	47	Scandinavian dictionary ordering
scannocp	48	Scandinavian case-insensitive dictionary ordering with preference
bin_utf8	50	UTF-8 binary sort order
dict	51	English dictionary, meaning: • accented characters are adjacent • accented characters are distinct • uppercase precedes lowercase
nocase	52	Dictionary, case-insensitive. Limited to ASCII only.
nocasep	53	Dictionary, case-insensitive, except in order by, where uppercase precedes lowercase.
noaccent	54	Dictinary, case- and accent-insensitive.
espdict	55	Spanish, dictionary
espnocs	56	Spanish, case insensitive
espnoac	57	Spanish, accent insensitive
rusnocs	59	Russian, case insensitive
cyrnocs	64	Common Cyrillic, case insensitive
elldict	65	Greek, dictionary
hundict	69	Hungarian, dictionary
hunnoac	70	Hungarian, accent insensitive
hunnocs	71	Hungarian, case insensitive

Name	ID	Description
turknoac	73	Turkish, accent insensitive
turknocs	74	Turkish, case insensitive

Table 7-7 lists the loadable sort orders.

Table 7-7: Loadable sort orders

Name	ID	Description
cp932bin	129	Japanese cp932
gb3213bn	137	Chinese gb2312
cyrdict	140	Cyrillic, dictionary
turdict	155	Turkish, dictionary
euckscbn	161	Korean euckcs
gbpinyin	163	Chinese gb2312 pinyin
rusdict	165	Russian, dictionary
sjisbin	179	Japanese, sjis binary
big5bin	194	Chinese b165

To view this sort order list in Adaptive Server, use the sp_helpsort system procedure. See Chapter 1, "System Procedures" in *Reference Manual: Procedures* for more information on sp_helpsort.

You can add sort orders using external files in the *$SYBASE/collate/Unicode* directory. The names and collation IDs are stored in SYSCHARSETS. The names of external Unicode sort orders do not have to be in SYSCHARSETS before you can set the default Unicode sort order.

Note External Unicode sort orders are provided by Sybase. Do not attempt to create external Unicode sort orders.

It is important to note that sort order associated with Unicode data is completely independent of the sort order associated with traditional character data. All relational expressions involving the Unicode datatypes will be performed using the Unicode sort order. This includes mixed-mode expressions involving Unicode and non-Unicode data. For example, in the following query the varchar character constant 'Mü' will be implicitly cast to unichar and the comparison will be performed according to the Unicode sort order:

```
select * from authors where unicode_name > 'Mü'
```

The same holds true for all other comparison operators, as well as the concatenation operator "+", the operator "in", and the operator "between." Once again, the goal is retain compatibility with existing database applications.

Tables joins based on equality (equi-joins) deserve special mention. These are generally optimized by the server to take advantage of indexes that defined on the participating columns. When a unichar column is joined with a char column, the latter will require a conversion, and since the character sort order and the Unicode sort order are distinct, the optimizer will ignore the index on the char column.

In Adaptive Server 12.5.1, when the server's default character set is configured to UTF-8, you can configure the server's default sort order (for char data) to be any of the above sort orders. Prior to this version, the binary sort order "bin_utf8" (ID=50) was the only well-behaved sort order for UTF-8. Although not required, the sort order for char data in UTF-8 can be selected so that it corresponds with the sort order for unichar.

There is a potential confusion regarding choice of binary sort orders for Unicode. The sort order named "binary" is the most efficient one for unichar data (UTF-16), and is thus the default. This order is based on the Unicode scalar value, meaning that all 32-bit surrogate pairs are placed after all 16-bit Unicode values. The sort order named "utf8bin" is designed to match the order of the default (most efficient) binary order for UTF-8 char data, namely "bin_utf8". The recommended matching combinations are thus "binary" for unichar and "binary" for UTF-8 char, or "utf8bin" for unichar and "bin_utf8" for UTF-8 char. The former favors unichar efficiency, while the latter favors char efficiency. The use of "utf8bin" for UTF-8 char is to be avoided, since it is equivalent to "bin_utf8" but less efficient.

Selecting a language for system messages

Any installation of Adaptive Server can use Language Modules containing files of messages in different languages. Adaptive Server provides Language Modules for messages in the following languages: English, Chinese (Simplified), French, German, Japanese, Korean, Brazilian Portuguese, and Spanish. If your client language is *not* one of these languages, you see system messages in English, the default language.

Each client can choose to view messages in their own language at the same time, from the same server; for example, one client views system messages in French, another in Spanish, and another in German. To do this, however, all selected languages *must* be part of the same language group. For example, French, Spanish and German are all part of language group 1. Japanese, on the other hand, is part of language group 101, which contains no other languages. Therefore, if Japanese is your server language, you can display system messages only in Japanese or English. Remember that *all* language groups can display messages in English. There is also a server-wide default language, used if the user has not selected a specific language. If you use Unicode, you can view system messages in any of the supported languages.

You can select the language for your system messages in one of two ways:

- Select a language as part of your user profile.
- Enter a language in the *locales.dat* file.

The following table displays the supported system message languages and their language groups. Each user can select only one language for system messages per session.

Table 7-8: Supported system messages

Language group	System message languages	Character sets
Group 1	French, German, Spanish, Brazilian Portuguese	ASCII 8, CP 437, CP 850, CP 860, CP 863, CP 1252, ISO 8859-1, ISO 8859-15, Macintosh Roman, ROMAN8
Group 101	Japanese	CP 932, DEC Kanji, EUC-JIS, Shift-JIS
Group 102	Simplified Chinese (PRC)	CP 936, EUC-GB
Group 104	Korean	EUC-KSC
Unicode	French, German, Spanish, Brazilian Portuguese, Japanese, Simplified Chinese, Korean	UTF-8
All Other Language Groups	English	

You need to install language modules for all languages in which clients will receive messages. These language modules, located in the *locales* subdirectory of the Adaptive Server installation directory, are part of a group of files called *localization files*. For information about localization files and the software message directory structure, refer to "Types of localization files" on page 294.

Setting up your server: examples

This section discusses setup options and the steps necessary to implement them. This is only a sample, and is meant to suggest ideas and methods for your own setup process.

A Spanish-version server

This situation discusses how to set up a new server with all clients using the same language. To do this:

1. Select the server language, in this case, Spanish. By reviewing Table 7-1 on page 264, you see that Spanish is part of language group 1. Based on your platform, select a character set from language group 1. Sybase recommends that you select the character set used by the greatest number of clients. Or, if you think your company might someday expand into other countries and languages, you might consider installing Unicode (see "Selecting the character set for your server" on page 263).
2. Install the Spanish language module in the server. This allows clients to view system messages in Spanish.
3. Select the default sort order. By referring to Table 7-5 on page 276, you see that Spanish has three possible sort orders, in addition to binary sort order. Select a sort order.
4. Restart the server.

A U.S.-based company in Japan

This situation involves clients in Japan, who will want to enter data, sort data, and receive system messages in Japanese, while submitting data to a server that is accessed by English-only users. To do this:

1. Select the default character set for your server. If you install a character set from language group 101 (Japanese), you can support both Japanese and English data in the same server.
2. Install the Japanese language module so that system messages are available in Japanese.

3 Select the sort order. By referring to Table 7-5 on page 276, you can see that a binary sort order is the only sort order available for Japanese. Therefore, both the English and Japanese clients will have a default binary sort order. Consider using the sortkey function to provide solutions for both audiences.

4 Make sure that each Japanese user requests Japanese messages by default. Since you are using a character set from language group 101, and you have already installed the Japanese language module, your client in Japan will see messages in Japanese, while clients in the U.S. can choose to see messages in English.

A Japan-based company with multinational clients

This company is located in Japan, and has clients in France, Germany, and Spain.

This situation indicates that you need to mix European and Asian languages in the same server.

1 Select the default server language and character set. Since your company is based in Japan and most of your clients are located in Japan, the default server language should be Japanese. But you also want your clients in France, Germany, and Spain to be able to send and receive data in their native languages. By reviewing Table 7-1 on page 264, you can see that Japanese is part of language group 101, while French, German, and Spanish are part of language group 1. Since the languages you need are not part of the same language group, the only way you can have all of these languages on the same server is to select Unicode as your default character set.

2 Install the language modules for Japanese, French, German, and Spanish.

3 Select the binary sort order, since this is the only sort order available for the Unicode character set. (You can, however, consider using the sortkey function inside your application code to supply data sorted according to each user’s preference.)

4 Select Japanese as the default language for system messages. Clients in other countries can select their own native language for messages.

Changing the character set, sort order, or message language

Even after you have configured your server, a System Administrator can change the default character set, sort order, or message language used by Adaptive Server. Because a sort order is built on a specific character set, changing character sets always involves a change in sort order. However, you can change the sort order without changing character sets, because more than one sort order may be available for a character set.

To display Adaptive Server's default sort order, character set, and a table of its primary sort orders, enter:

```
sp_helpsort
```

Changing the default character set

Adaptive Server can have only one *default character set*, the character set in which data is stored in its databases. When you install Adaptive Server, you specify a default character set.

Warning! Please read the following carefully and exercise caution when changing the default character set in Adaptive Server. Sybase strongly recommends that you perform backups before you change a default character set.

When you change the default character set in Adaptive Server, you need to convert any existing data to the new default character set. Conversion is unnecessary *only* if:

- There is no user data in the server.
- It is acceptable to destroy user data in the server.
- You are *absolutely certain* that data in the server uses only ASCll-7. In this case, you can change the default without first copying your data out of the server.

In all other cases, you must convert the existing data as follows:

1. Copy the data out using bcp.
2. Change the default character set.

3 Use bcp with the appropriate flags for data conversion to copy the data back into the server.

See the *Utility Guide* for more information about using bcp to copy data.

Warning! After converting data to a different character set (particularly to UTF-8), the data may be too large for the allocated column size. Re-create the columns affected with a larger size. Refer to the Unidb tool in the Sybase UDK product.

Code conversion between the character set of the existing data and the new default character set must be supported. If it is not, conversion errors will occur and the data will not be converted correctly. See Chapter 8, “Configuring Client/Server Character Set Conversions,” for more information about supported character set conversions.

Even if conversions are supported between the character sets, some errors may occur due to minor differences between the character sets, or because some characters do not have equivalents in other character sets. Rows containing problematic data may not get copied back into the database or data may contain partial or invalid characters.

Changing the default sort order

Adaptive Server can have only one *default sort order*, the collating sequence it uses to order data. When you consider changing the sort order for character data on a particular Adaptive Server, keep this in mind: all of your organization’s Adaptive Servers should have the same sort order. A single sort order enforces consistency and makes distributed processing easier to administer.

You may have to rebuild your indexes after changing the default sort order. For more information, see “Reconfiguring the character set, sort order, or message language” on page 284.

Reconfiguring the character set, sort order, or message language

This section summarizes the steps to take before and after changing Adaptive Server’s default character set, sort order, or message language. For procedures on how to configure the character set, sort order, or message language for a new server, see the configuration documentation for your platform.

If your data does not have to be converted to a new character set and both the old and the new character sets use binary sort order you can use a database dump. You can restore your database from backups that were made before the character set was reconfigured.

Note Back up all databases in Adaptive Server both before and after you change character sets or sort orders.

Usually, you cannot reload your data from a database dump when you have reconfigured the default character set and sort order.

If the following is true, use bcp to copy the data out of and into your databases.

- If a database contains character data, and you want the data to be converted to a new character set. Do not load a database dump of the data into an Adaptive Server with the new default character set. Adaptive Server interprets the data loaded as if it is in the new character set, and the data will be corrupted.
- If you are changing only the default sort order and not the default character set. You cannot load a database from a dump that was performed before you changed the sort order. If you attempt to do so, an error message appears, and the load is aborted.
- You change the default character set, and either the old or the new sort order is not binary. You cannot load a database dump that was made before you changed the character set.

Unicode examples

In the following example, a fictitious database named pubsx will be modified to use univarchar columns.

Schema

Assume a database was created using the following script on a server that has all the installation defaults, namely character set “iso_1” and default sortorder id 50, “binary_iso_1”.

```
> create database xpubs
> go
> use xpubs
> go
> create table authors (au_id int, au_lname
varchar(255), au_fname varchar(255))
> go
> create index au_idx on authors(au_lname, au_fname)
> go
```

Then the data was loaded into the server using a series of inserts and/or updates.

Converting to UTF-8

The first step towards using Unicode will be to extract the data and convert it to UTF-8 form.

```
% bcp xpubs..authors out authors.utf8.bcp -c -Jutf8 -Usa -P
```

The next step will be to install UTF-8 as the default character set in the server:

```
% charset -Usa -P binary.srt utf8
% isql -Usa -P
> sp_configure 'default sortorder id', 50, 'utf8'
> go
> shutdown
> go
```

Then reboot the server. It will modify the default character set, recreate indexes on the system tables, then exit. Reboot the server a second time. Then reload the data:

```
% isql -Usa -P
> sp_dboption xpubs, 'select into', true
> go
> use xpubs
> go
> checkpoint
> go
> delete from authors
> go
> quit

% bcp xpubs..authors in authors.utf8.bcp -c -Jutf8 -Usa -P
```

Converting selected columns to Unichar

With a working database running with UTF-8 as the default character set, it becomes a simple matter to convert select columns to univarchar:

```
% isql -Usa -P
```

```
> use xpubs
> go
> alter table authors modify au_lname univarchar(255),
au_fname univarchar(255)
> go
```

The columns will be modified to the new datatypes, the data will be converted in place, and the index will be re-created.

Preliminary steps

Before you run the installation program to reconfigure Adaptive Server:

1. Dump all user databases and the master database. If you have made changes to model or sybsystemprocs, dump them also.
2. Load the Language Module if it is not already loaded (see the configuration documentation for your platform for complete instructions).
3. If you are changing the Adaptive Server default character set, and your current databases contain non ASCII-7 data, use bcp to copy the existing data out of your databases.

Once you have loaded the Language Module, you can run the Adaptive Server installation program, which allows you to:

- Install or remove message languages and character sets included with Adaptive Server
- Change the default message language or character set
- Select a different sort order

See the configuration documentation for your platform for instructions on using the installation program.

To reconfigure the language, character set, or sort order, use the sqlloc utility, described in *Utility Guide for UNIX Platforms*. If you are using Windows NT, use the Server Config utility, described in *Configuration Guide for Windows NT*. If you are adding a new character set that is not included with Adaptive Server, see the *Sybase Character Sets* manual for complete instructions.

If you installed additional languages but did not change Adaptive Server's character set or sort order, you have completed the reconfiguration process.

If you changed the Adaptive Server default character set, and your current databases contain non ASCII-7 data, copy your data back into your databases, using bcp with the necessary flags to enable conversion.

If you changed Adaptive Server's default sort order or character set, see "Reconfiguring the character set, sort order, or message language" on page 284.

Setting the user's default language

If you install an additional language, users running client programs can run sp_modifylogin to set that language as their default language, or set the LANG variable on the client machine, with the appropriate entries in locales.dat.

Recovery after reconfiguration

Every time Adaptive Server is stopped and restarted, recovery is performed automatically on each database. Automatic recovery is discussed in detail in Chapter 27, "Developing a Backup and Recovery Plan."

After recovery is complete, the new sort order and character set definitions are loaded.

If you have changed the sort order, Adaptive Server switches to single-user mode to allow the necessary updates to system tables and to prevent other users from using the server. Each system table with a character-based index is automatically checked to see if any indexes have been corrupted by the sort order change. Character-based indexes in system tables are automatically rebuilt, if necessary, using the new sort order definition.

After the system indexes are rebuilt, character-based user indexes are marked "suspect" in the sysindexes system table, without being checked. User tables with suspect indexes are marked "read-only" in sysobjects to prevent updates to these tables and use of the "suspect" indexes until they have been checked and, if necessary, rebuilt.

Next, the new sort order information replaces the old information in the area of the disk that holds configuration information. Adaptive Server then shuts down so that it starts for the next session with a complete and accurate set of system information.

Using *sp_indsuspect* to find corrupt indexes

After Adaptive Server shuts down, restart it, and use sp_indsuspect to find the user tables that need to be reindexed. The following is the syntax, where *tab_name* is the optional name of a specific table:

sp_indsuspect [*tab_name*]

If *tab_name* is missing, sp_indsuspect creates a list of all tables in the current database that has indexes marked "suspect" when the sort order changes.

In this example, running sp_indsuspect in mydb database yields one suspect index:

```
sp_indsuspect
```

```
Suspect indexes in database mydb
Own.Tab.Ind (Obj_ID, Ind_ID) =
dbo.holdings.h_name_ix(160048003, 2)
```

Rebuilding indexes after changing the sort order

dbcc reindex checks the integrity of indexes on user tables by running a "fast" version of dbcc checktable. For details, see "dbcc checktable" on page 770. dbcc reindex drops and rebuilds the indexes where the sort order used is not consistent with the new sort order. When dbcc reindex discovers the first index-related error, it displays a message, and then rebuilds the inconsistent indexes. The System Administrator or table owner should run dbcc reindex after changing the sort order in Adaptive Server.

The syntax is:

dbcc reindex ({*table_name* | *table_id*})

Run this command on all tables listed by sp_indsuspect as containing suspect indexes. For example:

```
dbcc reindex(titles)
```

```
One or more indexes are corrupt. They will be rebuilt.
```

In the preceding example, dbcc reindex discovers one or more suspect indexes in the table titles; it drops and re-creates the appropriate indexes.

If the indexes for a table are already correct, or if there are no indexes for the table, dbcc reindex does not rebuild any indexes. It displays a message instead. If a table is suspected of containing corrupt data, the command is aborted. If that happens, an error message instructs the user to run dbcc checktable.

When dbcc reindex finishes successfully, all "suspect" marks on the table's indexes are removed. The "read-only" mark on the table is also removed, and the table can be updated. These marks are removed whether or not any indexes have to be rebuilt.

dbcc reindex does not reindex system tables. System indexes are checked and rebuilt, if necessary, as an automatic part of recovery after Adaptive Server is restarted following a sort order change.

Upgrading *text* data after changing character sets

If you have changed an Adaptive Server's character set to a **multibyte character set** use dbcc fix_text to upgrade text values.

The syntax is:

dbcc fix_text ({*table_name* | *table_id*})

Changing to a multibyte character set makes the management of text data more complicated. A text value can be large enough to cover several pages; therefore, Adaptive Server must be able to handle characters that span page boundaries. To do so, Adaptive Server requires additional information on each of the text pages. The System Administrator or table owner must run dbcc fix_text on each table that has text data to calculate the new values needed.

To see the names of all tables that contain text data, use:

```
select sysobjects.name
from sysobjects, syscolumns
where syscolumns.type = 35
and sysobjects.id = syscolumns.id
```

The System Administrator or table owner must run dbcc fix_text to calculate the new values needed.

The syntax of dbcc fix_text is:

dbcc fix_text (*table_name* | *table_id*)

The table named must be in the current database.

dbcc fix_text opens the specified table, calculates the character statistics required for each text value, and adds the statistics to the appropriate page header fields. This process can take a long time, depending on the number and size of the text values in a table. dbcc fix_text can generate a large number of log records, which may fill up the transaction log. dbcc fix_text performs updates in a series of small transactions so that if a log becomes full, only a small amount of work is lost.

If you run out of log space, clear out your log (see Chapter 28, “Backing Up and Restoring User Databases”). Then restart dbcc fix_text, using the same table that was being upgraded when the original dbcc fix_text halted. Each multibyte text value contains information that indicates whether it has been upgraded, so dbcc fix_text upgrades only the text values that were not processed in earlier passes.

If your database stores its log on a separate segment, you can use thresholds to manage clearing the log. See Chapter 31, “Managing Free Space with Thresholds.”

If dbcc fix_text cannot acquire a needed lock on a text page, it reports the problem and continues with the work, like this:

```
Unable to acquire an exclusive lock on text page 408.
This text value has not been recalculated.  In order to
recalculate those TEXT pages you must release the lock
and reissue the dbcc fix_text command.
```

Retrieving *text* values after changing character sets

If you attempt to retrieve text values after changing to a multibyte character set, and you have not run dbcc fix_text, the command fails with this error message:

```
Adaptive Server is now running a multi-byte character
set, and this TEXT column’s character counts have not
been recalculated using this character set. Use dbcc
fix_text before running this query again.
```

Note If you have changed the sort order or character set and errors occurred, see “How to Manually Change Sort Order or Default Character Set” in the *Adaptive Server Enterprise Troubleshooting and Error Messages Guide*.

Installing date strings for unsupported languages

You can use sp_addlanguage to install names for the days of the week and months of the year for languages that do not have Language Modules. With sp_addlanguage, you define:

- A language name and (optionally) an alias for the name

- A list of the full names of months and a list of abbreviations for the month names
- A list of the full names of the days of the week
- The date format for entering dates (such as month/day/year)
- The number of the first day of the week

This example adds the information for Italian:

```
sp_addlanguage italian, italiano,
"gennaio,febbraio,marzo,aprile,maggio,giugno,luglio,agosto,settembre,ottobre,
novembre,dicembre",
"genn,feb,mar,apr,mag,giu,lug,ago,sett,ott,nov,dic",
"lunedi,martedi,mercoledi,giovedi,venerdi,sabato,domenica",
dmy, 1
```

sp_addlanguage enforces strict data entry rules. The lists of month names, month abbreviations, and days of the week must be comma-separated lists with no spaces or line feeds (returns). Also, they must contain the correct number of elements (12 for month strings, 7 for day-of-the-week strings.)

Valid values for the date formats are: mdy, dmy, ymd, ydm, myd, and dym. The dmy value indicates that the dates are in day/month/year order. This format affects only data entry; to change output format, you must use the convert function.

Server versus client date interpretation

Generally, date values are resolved on the client. When a user selects date values, Adaptive Server sends them to the client in internal format. The client uses the *common.loc* file and other localization files in the default language subdirectory of the *locales* directory on the client to convert the internal format to character data. For example, if the user’s default language is Spanish, Adaptive Server looks for the *common.loc* file in */locales/spanish/char_set*. It uses the information in the file to display, for example, 12 febrero 1997.

Assume that the user’s default language is set to Italian, a language for which Adaptive Server does not provide a Language Module, and that the date values in Italian have been added. When the client connects to the server and looks for the *common.loc* file for Italian, it does not find the file. The client prints an error message and connects to the server. If the user then selects date values, the dates are displayed in U.S. English format.To display the date values added with sp_addlanguage, use the convert function to force the dates to be converted to character data at the server.

The following query generates a result set with the dates in U.S. English format:

```
select pubdate from titles
```

The query below, however, returns the date with the month names in Italian:

```
select convert(char(19),pubdate) from titles
```

Internationalization and localization files

Types of internationalization files

The files that support data processing in a particular language are called *internationalization files*. Several types of internationalization files come with Adaptive Server. Table 7-9 describes these files.

Table 7-9: Internationalization files

File	Location	Purpose and contents
charset.loc	In each character set subdirectory of the *charsets* directory	Character set definition files that define the lexical properties of each character, such as alphanumeric, punctuation, operand, and uppercase or lowercase. Used by Adaptive Server to correctly process data.
**.srt*	In each character set subdirectory of the *charsets* directory	Defines the sort order for alphanumeric and special characters, including ligatures, diacritics, and other language-specific considerations.
**.xlt*	In each character set subdirectory of the *charsets* directory	Terminal-specific character translation files for use with utilities such as bcp and isql. For more information about how the *.xlt* files are used, see Chapter 8, "Configuring Client/Server Character Set Conversions," and the *Utility Guide*.

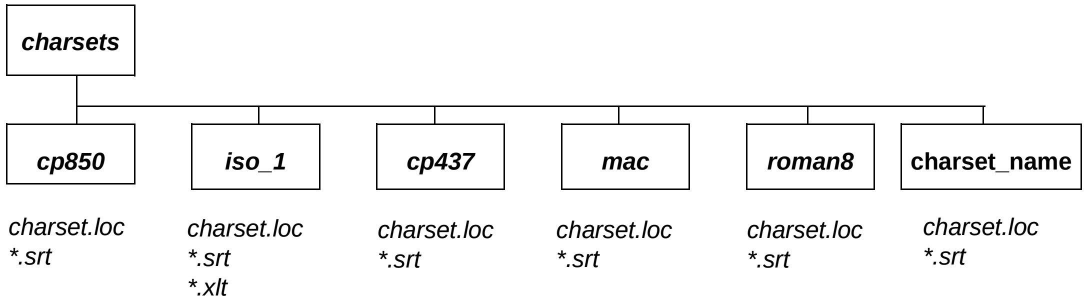
Warning! Do not alter any of the internationalization files. If you need to install a new terminal definition or sort order, contact your local Sybase office or distributor.

Character sets directory structure

Figure 7-3 shows the directory structure for the Western European character sets that come with Adaptive Server. There is a separate subdirectory for each character set in the *charsets* directory. Within the subdirectory for each character set (for example, *cp850*) are the character set and sort order definition files and terminal-specific files.

If you load additional character sets, they will also appear in the *charsets* directory:

Figure 7-3: Structure of the charsets directory



The following global variables contain information about character sets:

@@char_convert	Contains 0 if character set conversion is not in effect. Contains 1 if character set conversion is in effect.
@@client_csname	The client's character set name. Set to NULL if client character set has never been initialized; otherwise, it contains the name of the character set for the connection.
@@client_csid	The client's character set ID. Set to -1 if client character set has never been initialized; otherwise, it contains the client character set ID from syscharsets for the connection.
@@maxcharlen	The maximum length, in bytes, of a character in Adaptive Server's default character set.
@@ncharsize or *@@charsize*	The maximum length, in bytes, of a character set in the current server default character set.
@@unicharsize	Equals 2.

Types of localization files

Adaptive Server includes several localization files for each Language Module, as shown in Table 7-10.

Table 7-10: Localization files

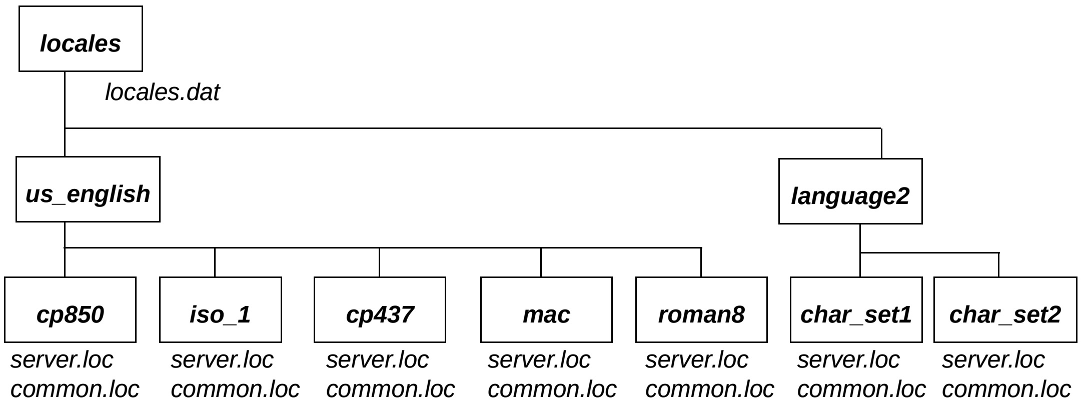
File	Location	Purpose and contents
locales.dat	In the *locales* directory	Used by client applications to identify the default message language and character set.
server.loc	In the character set subdirectories under each language subdirectory in the *locales* directory	Software messages translated into the local language. Sybase products have product-specific **.loc* files. If an entry is not translated, that software message or string appears in U.S. English instead of the local language.
common.loc	In each language and character set directory of the *locales* directory	Contains the local names of the months of the year and their abbreviations and information about the local date, time, and money formats.

Warning! Do not alter any of the localization files. If you need to alter any information in those files, contact your local Sybase office or distributor.

Software messages directory structure

Figure 7-4 shows how localization files are arranged. Within the *locales* directory is a subdirectory for each language installed. There is always a *us_english* subdirectory. (On PC platforms, this directory is called *english*.) During installation, when you are prompted to select the languages you want installed on Adaptive Server, the install program lists the supported software message languages. If you install Language Modules for additional languages, you will see subdirectories for those languages. Within each language are subdirectories for the supported character sets; for example, *cp850* is a supported character set for *us_english*. Software message files for each Sybase product reside in the character set subdirectories.

Figure 7-4: Messages directory structure



Message languages and global variables

The following global variables contain information about languages:

@@langid	Contains the local language ID of the language currently in use (specified in syslanguages.langid)
@@language	Contains the name of the language currently in use (specified in syslanguages.name)

CHAPTER 8

Configuring Client/Server Character Set Conversions

This chapter describes how to configure character set conversion when the client uses a different character set than Adaptive Server.

Character set conversion in Adaptive Server

In a heterogeneous environment, Adaptive Server may need to communicate with clients running on different platforms using different character sets. Although different character sets may support the same language group (for example, ISO 8858-1 and CP 850 support the group 1 languages), they may encode the same characters differently. For example, in ISO 8859-1, the character *à* is encoded as *0xE0* in hexadecimal. However, in CP 850 the same character is encoded as *0x85* in hexadecimal.

To maintain data integrity between your clients and servers, data must be converted between the character sets. The goal is to ensure that an “a” remains an “a” even when crossing between machine and character set boundaries. This process is known as *character set conversion*.

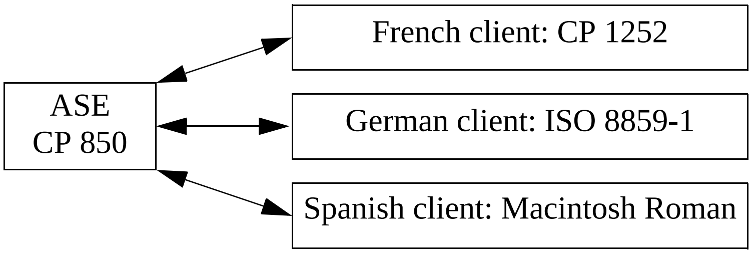
Supported character set conversions

Character set conversion occurs between a pair of character sets. The supported conversions in any particular client/server system depend on the character sets used by the server and its clients. One type of character set conversion occurs if the server uses a native character set as the default; a different type of conversion is used if the server default is Unicode UTF-8.

Conversion for native character sets

Adaptive Server supports character set conversion between native character sets belonging to the same language group. If the server has a native character set as its default, the clients' character sets must belong to the same language group. Figure 8-1is an example of a Western European client/server system. In this example, the clients' character sets and Adaptive Server's default character set all belong to Group 1. Data is correctly converted between the client character sets and the server default character set. Since they all belong to the same language group, the clients can view all data on the server, no matter which client submitted the data.

Figure 8-1: Character set conversion when server and client character sets belong to the same language group

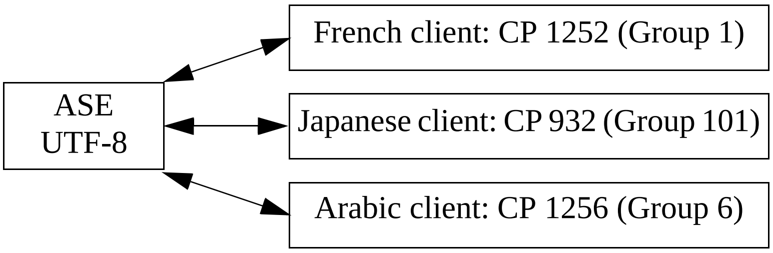


For a list of the language groups and supported character sets, see Table 7-1 on page 264.

Conversion in a Unicode system

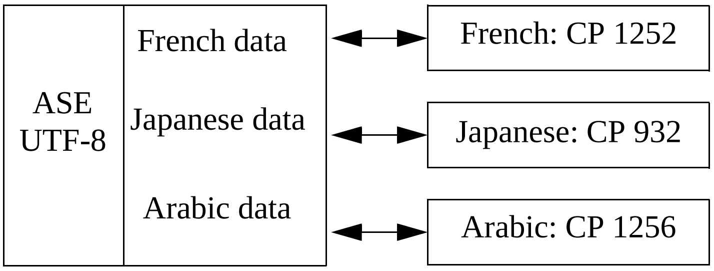
Adaptive Server also supports character set conversion between UTF-8 and any native character set that Sybase supports. In a Unicode system, since the server default character set is UTF-8, the client character set may be a native character set from any language group. Therefore, a Japanese client (group 101), a French client (Group 1), and an Arabic client (group 6) can all send and receive data from the same server. Data from each client is correctly converted as it passes between each client and the server.

Figure 8-2: Character set conversion in a Unicode system


ASE
UTF-8
French client: CP 1252 (Group 1)
Japanese client: CP 932 (Group 101)
Arabic client: CP 1256 (Group 6)

Note however, that each client can view data only in the language supported by its character set. Therefore, the Japanese client can view any Japanese data on the server, but it cannot view Arabic or French data. Likewise, the French client can view French or any other Western European language supported by its character set, but not Japanese or Arabic.

Figure 8-3: Viewing Unicode data



An additional character set, ASCII 7, is a subset of *every* character set, including Unicode, and is therefore compatible with all character sets in all language groups. If either the Adaptive Server or the client's character set is ASCII 7, any 7-bit ASCII character can pass between the client and server unaltered and without conversion.

Sybase does not recommend that you configure a server for ASCII-7, but you can achieve the same benefits of compatibility by restricting each client to use only the first 128 characters of each native character set.

Types of character set conversion

Character set conversion is implemented on Adaptive Server in two different ways:

- Adaptive Server direct conversions
- Unicode conversions

Adaptive Server direct conversions

Adaptive Server direct conversions support conversions between two native character sets of the *same* language group. For example, Adaptive Server supports conversion between CP 437 and CP 850, because both belong to the group 1 language group. Adaptive Server direct conversions exist between many, but not all, native character sets of a language group (see Table 8-1 on page 302).

Unicode conversions

Unicode conversions exists for all native character sets. When converting between two native character sets, Unicode conversion uses Unicode as an intermediate character set. For example, to convert between the server default character set (CP 437), and the client character set (CP 860), CP 437 is first converted to Unicode; Unicode is then converted to CP 860.

$$\begin{tikzcd}\text{CP 437} \arrow[r] & \text{Unicode} \arrow[r] & \text{CP 860}\end{tikzcd}$$

As this example illustrates, Unicode conversions may be used either when the default character set of the server is UTF-8, or a native character set. You must specifically configure your server to use Unicode conversions (unless the server’s default character set is UTF-8).

Earlier versions of Adaptive Server used direct conversions, and it is the default method for character set conversions. However, Unicode conversions implemented in more recent versions of Adaptive Server releases allow easier and less complex character set conversion. Sybase continues to support existing Adaptive Server direct conversions, but Sybase now also uses Unicode conversions to provide complete conversion support for all character sets. Sybase has no plans to add new direct conversions.

Which type of conversion do I use?

To determine the conversion options that are available for your client/server system, see Table 8-1 on page 302.

Non-Unicode client/server systems

In a non-Unicode system, the character sets of the server and clients are native character sets; therefore, you can use the Adaptive Server direct conversions.

However, there are some character sets for which there is no Adaptive Server direct conversion; in this situation, you must use Unicode conversions.

- If all character sets in your client/server system fall into column 1 of Table 8-1, use the Adaptive Server direct conversions. The character sets must all belong to the same language group.
- If the character sets in your client/server system fall into column 2 of Table 8-1, or some combination of columns 1 and 2, then you *must* configure your server to use Unicode conversions. Again, the character sets must all belong to the same language group.

For example, assume the server default character set is CP 850 and the clients' character sets are either ISO 8859-1 or ROMAN 8. Table 8-1 shows that direct conversions exist between CP 850 and the client character sets. Now, suppose you add a client using CP 1252 to this configuration. Since there is no direct conversion between CP 1252 and CP 850, (the default server character set), you *must* use Unicode conversions to convert between CP 1252 and CP 850. When you have a mixture of character sets—some where you can use Adaptive Server direct conversions and others where you must use Unicode conversions—you can specify that a combination of Adaptive Server direct conversion and Unicode conversion be used.

Unicode client/server systems

If your server default is Unicode UTF-8, then all conversions are between UTF-8 and whatever native character set is being used on the client systems. Therefore, in a Unicode system, Unicode conversions are used *exclusively*.

Table 8-1: Conversion methods for character sets

Language group	Column 1 – Adaptive Server direct conversions and Unicode conversions	Column 2 – Unicode conversions only
Group 1	CP 437, CP 850, ISO 8859-1, Macintosh Roman, ROMAN8	CP 860, CP 1252, ISO 8859-15, CP 863
Group 2	CP 852, CP 1250, CP 8859-1, Macintosh Central European	ISO 8859-2
Group 4	No conversions needed (only one character set supported)	
Group 5	CP 855, CP 866, CP 1251, ISO 8859-5, Koi8, Macintosh Cyrillic	
Group 6		CP 864, CP 1256, ISO 8859-6
Group 7	CP 869, CP 1253, GREEK8, ISO 8859-7, Macintosh Greek	
Group 8		CP 1255, ISO 8859-8
Group 9	CP 857, CP 1254, ISO 8859-9, Macintosh Turkish, TURKISH8	
Group 101	DEC Kanjii, EUC-JIS, Shift-JIS	CP 932
Group 102		CP 936, EUG-GB
Group 103		Big 5, CP 950, EUC-CNS
Group 104	No conversions needed (only one character set supported)	
Group 105		CP 874, TIS 620
Group 106	No conversions needed (only one character set supported)	
Unicode	No conversions needed (only one character set supported)	

Configuring the server

By default, Adaptive Server uses direct conversions to convert data between different character sets. To use the Unicode conversions, you must configure the server with the sp_configure command. Set the enable unicode conversions option to either 1 or 2.

- If you set sp_configure "enable unicode conversions" to 1:

  This setting uses Adaptive Server direct conversions or Unicode conversions. Adaptive Server first checks to see if an Adaptive Server direct conversion exists for the server and client character set. If direct conversion, it uses the direct conversion exists, it is used; if no direct conversion exists, the Unicode conversion is used.

Use this setting if the character sets in your client/server system fall into both columns 1 and 2 in Table 8-1.

- If you set sp_configure "enable unicode conversions" to 2:

  This setting uses Unicode conversions *only*. Adaptive Server uses Unicode conversions, without attempting to find an Adaptive Server direct conversion.

  Use this setting if the client/server conversions result in a change in the data length (see "Conversions and changes to data lengths" on page 305)

If all character sets fall into column 2 in Table 8-1, then you should set enable unicode conversions to 2 to always use Unicode conversions.

If the server default is UTF-8, the server automatically uses Unicode conversions only.

Enabling and disabling character set conversion

When a client requests a connection, the client identifies its character set to Adaptive Server. Adaptive Server compares the client character set with its default character set, and if the two names are identical, no conversion is required. If the names differ, Adaptive Server determines whether it supports conversion between its default and the client's character set. If it does not, it send an error message to the client and continues with the logon process. If it does, then character set conversion is automatically enabled. If the default character set of the server is UTF-8, it automatically uses Unicode conversions. If the default is a native character set, the server uses ASE direct conversions, unless the user specifies that Unicode conversions be used.

You can disable character set conversion at the server level. You may want to do this if:

- All of your clients are using the same character set as the server default, and therefore, no conversion is required.
- Conversion between the client character set and the server default is not supported.
- You want to store data in the server without converting the data, that is, without changing the encoding of the data.

To disable character set conversion at the server level, set the disable character set conversion parameter to 1. No conversion will occur for any client connecting to the server. By default this parameter is set to 0, which enables conversions.

You can also control character set conversion at the connection level using the set char_convert command from within a client session. set char_convert off turns conversion off between a particular client and the server. You may want to set char_convert off if the client and the server use the same character set, which makes conversion unnecessary. set char_convert on turns conversion back on.

Characters that cannot be converted

During the conversion process, some characters may not be converted. Here are two reasons:

- The character exists (is encoded) in the source character set, but it does not exist in the target character set. For example, the OE ligature, is part of the Macintosh character set (code point 0xCE). This character does not exist in the ISO 8859-1 character set. If the OE ligature exists in data that is being converted from the Macintosh to the ISO 8859-1 character set, it causes a conversion error.
- The character exists in both the source and the target character set, but in the target character set, the character is represented by a different number of bytes than in the source character set.

  For example, 1-byte accented characters (such as á, è) are 2-byte characters in UTF-8; 2-byte Thai characters are 3-byte characters in UTF-8. You can avoid this limitation by configuring the enable unicode conversion option to 1 or 2.

Error handling in character set conversion

Adaptive Server's character set conversion reports conversion errors when a character exists in the client's character set but not in the server's character set, or vice versa. Adaptive Server must guarantee that data successfully converted on input to the server can be successfully converted back to the client's character set when the client retrieves that data. To do this effectively, Adaptive Server must avoid putting suspect data into the database.

When Adaptive Server encounters a conversion error in the data being entered, it generates this message:

```
Msg 2402, Severity 16 (EX_USER):
Error converting client characters into server's
character set. Some character(s) could not be converted.
```

A conversion error prevents query execution on insert and update statements. If this occurs, review your data for problem characters and replace them.

When Adaptive Server encounters a conversion error while sending data to the client, it replaces the bytes of the suspect characters with ASCII question marks (?). However, the query batch continues to completion. When the statement is complete, Adaptive Server sends the following message:

```
Msg 2403, Severity 16 (EX_INFO):
WARNING! Some character(s) could not be converted into
client's character set. Unconverted bytes were changed
to question marks ('?').
```

Conversions and changes to data lengths

In some cases, converting data between the server's character set and the client's character set results in a change to the length of the data. For example, this occurs when the character set on one system uses one byte to represent each character and the character set on the other system requires two bytes per character.

When character set conversion results in a change in data length, there are two possibilities:

- The data length decreases, as in the following examples:
    - Greek or Russian in multibyte UTF-8 to a single-byte Greek or Russian character set

  - Japanese two-byte Hankaku Katakana characters in EUC-JIS to single-byte characters in Shift-JIS
- The data length increases, as in the following examples:
  - Single-byte Thai to multibyte Thai in UTF-8
  - Single-byte Japanese characters in Shift-JIS to two-byte Hankaku Katakana in EUC-JIS

Configuring your system and application

If you are using UTF-8 anywhere in your client/server system, or using a Japanese character set, you are likely to encounter changes in data length as a result of character set conversion. If either of these conditions is true, you must configure your server to handle changes in data length. You may also need to set up your client to handle changes in data length.

1. Configure the server to use Unicode conversions. See “Configuring the server” on page 302. If the data length increases between the server and the client, then you must also complete steps 2 and 3.
2. The client must be using Open Client 11.1 or later. It must inform the server that it is able to handle CS_LONGCHAR data at connection time, using the Open Client ct_capability function.

   The *capability* parameter must be set to CS_DATA_LCHAR and the *value* parameter must be set to CS_TRUE, where *connection* is a pointer to a CS_CONNECTION structure:

   ```
   CS_INT capval = CS_TRUE
   ct_capability(connection,CS_SET,CS_CAP_RESPONS,
        CS_DATA_LCHAR,&capval)
   ```
3. When conversions result in an increase in data length, char and varchar data are converted to the client’s character set and are sent to the client as CS_LONGCHAR data. The client application must be coded to extract the data received as CS_LONGCHAR.

Specifying the character set for utility programs

The Sybase utility programs assume that the default character set of the client platform is the same character set the client is using. However, sometimes the client character set differs from the character set for the platform. For this reason, you may need to specify the client character set at the command line. Character set conversion can be controlled in the standalone utilities. A command line option for the isql, bcp, and defncopy utilities specifies the client's character set and temporarily overrides settings of the LANG variable or settings in *locales.dat*.

-J *charset_name* (UNIX and PC) sets the client's character set to the *charset_name*.

Omitting the client character set's command-line flag causes the platform's default character set to be used. See the *Utility Guide* for information.

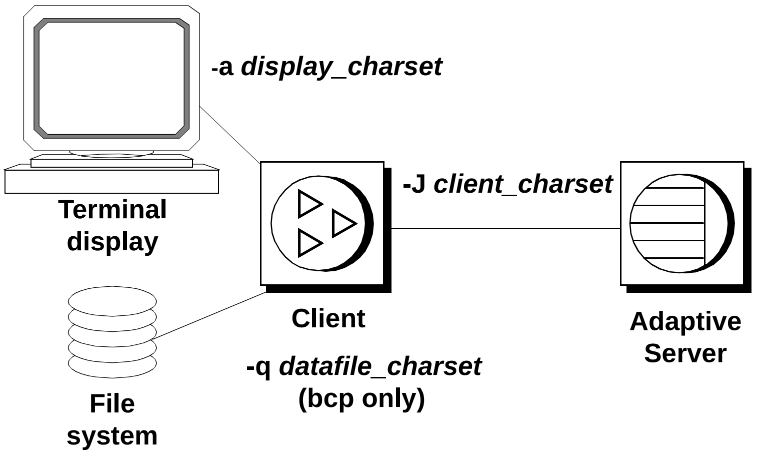
Display and file character set command-line options

Although the focus of this chapter is on character set conversion between client and Adaptive Server, there are two other places where you may need character set conversion:

- Between the client and a terminal
- Between the client and a file system

Figure 8-4 illustrates the paths and command-line options that are available in the standalone utilities isql, bcp, and defncopy.

Figure 8-4: Where character set conversion may be needed



As described earlier, the -J or /clientcharset command-line option specifies the character set used by the client when it sends and receives character data to and from Adaptive Server.

Setting the display character set

Use the -a command-line option if you are running the client from a terminal with a character set that differs from the client character set. In Figure 8-4, the -a option and the -J option are used together to identify the character set translation file (*.xlt* file) needed for the conversion.

Use -a without -J only if the client character set is the same as the default character set.

Setting the file character set

Use the -q command-line option if you are running bcp to copy character data to or from a file system that uses a character set that differs from the client character set. In Figure 8-4, use the -q or /filecharset option and the -J or /clientcharset option together to identify the character set translation file (*.xlt* file) needed for the conversion.

CHAPTER 9 Security Administration

This chapter provides an overview of the security features available in Adaptive Server.

Security features available in Adaptive Server

SQL Server version 11.0.6 passed the security evaluation by the National Security Agency (NSA) at the Class C2 criteria. (The requirements for the C2 criteria are given by the Department of Defense in DOD 52.00.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* [TCSEC], also known as the "Orange Book.")

The configuration of SQL Server version 11.0.6 that was evaluated at the C2 security level by the NSA in 1996 on the HP 9000 HP-UX BLS, 9.09+ platform is referred to as the evaluated configuration. Certain features of SQL Server, such as remote procedures and direct updates to system tables, were excluded from the evaluated configuration. Notes in the Adaptive Server documentation indicate particular features that were not included in the evaluated configuration. For a complete list of features that were excluded from the evaluated configuration, see Appendix A in the *SQL Server Installation and Configuration Guide for HP 9000 HP-UX BLS, 9.09+*.

Adaptive Server contains all of the security features included in SQL Server version 11.0.6 plus some new security features. Table 9-1 summarizes the major features.

Table 9-1: Major security features

Security feature	Description
Discretionary Access Controls (DAC)	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 upon an object owner's discretion.
Identification and authentication controls	Ensures that only authorized users can log into the system.
Division of roles	Allows you to grant privileged roles to specified users so that 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."
Network-based security	Provides security services to authenticate users and protect data transmitted among machines on a network.
Auditing	Provides the capability 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. In addition, Adaptive Server provides a single option to audit a set of server-wide security-relevant events.

General process of security administration

Table 9-2 describes the major tasks that are required to administer Adaptive Server in a secure manner and refers you to the documentation that contains the instructions for performing each task.

Table 9-2: General process for security administration

Task	Description	See
1. Install Adaptive Server, including auditing.	This task includes preparing for installation, loading files from your distribution medium, performing the actual installation, and administering the physical resources that are required.	The the installation documentation for your platform
2. Set up a secure administrative environment.	This includes enabling auditing, granting roles to individual users to ensure individual accountability, and assigning login names to System Administrators and System Security Officers.	Chapter 10, "Managing Adaptive Server Logins, Database Users, and Client Connections"

Task	Description	See
3. Add user logins to the server; add users to databases; establish groups and roles; set proxy authorization.	Add logins, create groups, add users to databases, drop and lock logins, and assign initial passwords. Assign roles to users, create user-defined roles, and define role hierarchies and mutual exclusivity of roles.	Chapter 10, "Managing Adaptive Server Logins, Database Users, and Client Connections"
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.	Chapter 12, "Managing User Permissions"
5. Administer the use of remote servers.	Establish and administer the access that is permitted between servers, add and drop remote server access, and map remote login names to local login names.	Chapter 13, "Managing Remote Servers," and the Adaptive Server installation and configuration documentation for your platform
6. 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.	Chapter 11, "Auditing," and the Adaptive Server installation and configuration documentation for your platform
7. Set up your installation for network-based security services.	Configure the server to use services, such as unified login, data confidentiality with encryption, data integrity, and determine security for remote procedures.	Chapter 14, "Using Kerberos, DCE, and Windows NT LAN Manager"

Guidelines for setting up security

Use the guidelines described in the following sections when you set up security on Adaptive Server.

Using the "sa" login

When Adaptive Server is installed, a single login called "sa" is configured with the System Administrator and System Security Officer roles. This means that the "sa" login has unlimited power.

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.

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.

Changing the "sa" login password

The "sa" login is configured initially with a "NULL" password. Use sp_password to change the password immediately after installation.

When to enable 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

Suppose you have decided to assign special roles to the users listed in Table 9-3.

Table 9-3: Users to whom you will assign roles

Name	Role	Operating system login name
Rajnish Smith	sso_role	rsmith
Catharine Macar-Swan	sa_role	cmacar
Soshi Ikedo	sa_role	sikedo
Julio Rozanski	oper_role	jrozan

Table 9-4 shows the sequence of commands you might use to set up a secure operating environment for Adaptive Server, based upon the role assignments shown in Table 9-3. After logging in to the operating system, you would issue these commands using the initial "sa" account.

Table 9-4: Examples of commands used to set up security

Commands	Result
• isql -Usa	Logs in to Adaptive Server as "sa". Both sa_role and sso_role are active.
• sp_audit "security", "all", "all", "on"' • sp_audit "all", "sa_role", "all", "on" • sp_audit "all", "sso_role", "all", "on"'	Sets auditing options for server-wide, security-relevant events and the auditing of all actions that have sa_role or sso_role active.
• sp_configure "auditing", 1	Enables auditing.

Note Before you enable auditing, set up a threshold procedure for the audit trail and determine how to handle the transaction log in sybsecurity. For details, see Chapter 11, "Auditing."

Commands	Result
• sp_addlogin rsmith, js&2P3d, @fullname = "Rajnish Smith" • sp_addlogin cmacar, Fr3ds#1, @fullname = "Catharine Macar-Swan" • sp_addlogin sikedo, mi5pd1s, @fullname = "Soshi Ikedo" • sp_addlogin jrozan, w1seCrkr, @fullname = "Julio Rozanski"	Adds logins and passwords for Rajnish, Catharine, Soshi, and Julio. A default database is not specified for any of these users, so their default database is master.
• grant role sso_role to rsmith • grant role sa_role to sikedo • grant role sa_role to cmacar • grant role oper_role to jrozan	Grants the sso_role to Rajnish, the sa_role to Soshi and Catharine, and the oper_role to Julio.
• use sybsecurity • sp_changedbowner rsmith	Grants access to the auditing database, sybsecurity, by making Rajnish, who is the System Security Officer, the database owner.

Commands	Result
sp_locklogin sa,"lock"	Locks the “sa” login so that no one can log in as “sa”. Individuals can assume only the roles that are configured for them.

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.

Discretionary access controls

Owners of objects can grant access to those objects to other users. Object owners can also grant other users the ability to pass the access permission to other users. With Adaptive Server’s discretionary access controls, you can give various kinds of permissions to users, groups, and roles with 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, views, 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, Database Owner, or database object owner), and by whether or not a particular user has been granted a permission with the option to grant that permission to other users.

Discretionary access controls are discussed in Chapter 12, “Managing User Permissions.”

Identification and authentication controls

Each Adaptive Server user is given a login account with a unique ID. All of that user’s activity on the server can be attributed to a server user ID and audited.

Adaptive Server passwords are stored in the master..syslogins table in encrypted form. When you log into Adaptive Server from a client, you can choose client-side password encryption to encrypt your password before sending it over the network.

A System Security Officer can grant a user the ability to impersonate another user in the server. This ability, called **proxy authorization**, allows administrators to check permissions for a particular user or to perform maintenance on a user’s database objects. Application servers can log in to the server and execute procedures and commands on behalf of several users.

Identification and authentication controls with network based security

Adaptive Server allows users to be pre-authenticated by a security mechanism before they log in to the server. This capability, called **unified login**, enables a user to log in to several servers without having to supply a login name and password for every connection.

Identification and authentication controls are discussed in Chapter 10, “Managing Adaptive Server Logins, Database Users, and Client Connections.” In addition, see “Using proxy authorization” and Chapter 13, “Managing Remote Servers.”

Division of roles

An important feature in Adaptive Server is the 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 provide individual accountability for users performing operational and administrative tasks. Their actions can be audited and attributed to them.

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 Analyst can perform all tasks and see all data that can be viewed by the Salary Administrators and Financial Analysts.

Mutual exclusivity

Two roles can be defined to be mutually exclusive for:

- Membership – a single user cannot be granted both roles. For example, an installation might not want a single user to have both the “payment_requestor” and “payment_approver” roles to be granted to the same user.
- Activation – a single user cannot activate, or enable, both roles. For example, a user might be granted both the “senior_auditor” and the “equipment_buyer” roles, but the installation may not want to permit the user 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 “Tech Support” roles. In addition, you might 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 356 for information on administering and using roles.

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, are 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.

Certificates

Certificates are like passports: once you have been assigned one, the authorities have all your identification information in the system. Immigration control can access your information as you travel from country to country. 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 log in 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).

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 CipherSuites, 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 CipherSuites that Adaptive Server supports, see “CipherSuites” on page 329.

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.

Adaptive Server uses the SSL Plus™ library API from Certicom Corp.

SSL filter

Adaptive Server’s directory service, such as the *interfaces* file, NT 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 that multiple network and security protocols can be enabled for a single server. Server connection attributes are specified with directory services, such as LDAP or DCE, or with the traditional Sybase *interfaces* file. See “Creating server directory entries” on page 326.

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:

```
SYBSRV1
    master tli tcp /dev/tcp \x00020abc123456780000000000000000 ssl
    query tli tcp /dev/tcp \x00020abc123456780000000000000000 ssl
    master tli tcp /dev/tcp \x00020abd123456780000000000000000
```

The SSL filter is different from other security mechanisms, such as DCE and 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.

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*

NT *%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*

- NT – *%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 325.

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, then 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).

v **Enabling SSL**

1 Generate a certificate for the server.

2 Create a trusted roots file.

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 disables SSL. This value is the default.

4 Add the SSL filter to the *interfaces* file. See “Creating server directory entries” on page 326.

5 Use sp_ssladmin to add a certificate to the certificates file. See “Administering certificates” on page 327.

6 Shut down and restart Adaptive Server.

Note To request, authorize, and convert third-party certificates, see the *Utility Guide* for information on certauth, certreq, and certpk12 tools.

Unlike other security services, such as DCE, Kerberos, and NTLAN, SSL relies neither on the “Security” section of the Open Client/Open Server configuration file *libtcl.cfg,* nor objects in *objectid.dat.*

The System Administrator should consider memory use by SSL when planning for total physical memory. You will 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 CA. 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.

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.

Third-party tools to request certificates

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* for your platform.

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.

For information about Sybase utilities, certauth, certreq, and certpk12 for requesting, authorizing and converting third-party certificates, see the *Utility Guide* for your platform.

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.

Adaptive Server supports the Certicom Corp. cryptographic engine, Security Builder™, which supports RSA and DSA algorithms to construct the certificate requests.

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 DCE or Kerberos are identified with a SECMECH line. The following example shows a TLI-based entry for an Adaptive Server using SSL in a UNIX environment:

An entry for an Adaptive Server with SSL and DCE security mechanisms on UNIX might look like:

```
SYBSRV1
    master tli tcp /dev/tcp \x00020abc12345678000000000000000000 ssl
    query tli tcp /dev/tcp \x00020abc1234567800000000000000000 ssl
    master tli tcp /dev/tcp \x00020abd12345678000000000000000000
    SECMECH 1.3.6.1.4.897.4.6.1
```

An entry for the server with SSL and Kerberos security mechanisms on NT might look like:

```
[SYBSRV2]
    query=nlwnsck, 18.52.86.120,2748,ssl
    master=nlwnsck 18.52.86.120,2748,ssl
    master=nlwnsck 18.52.86.120,2749
    secmech=1.3.6.1.4.897.4.6.6
```

The SECMECH lines for SYBSRV1 and SYBSRV2 in the examples contain an object identifier (OID) that refers to security mechanisms DCE and Kerberos, respectively. The OID values are defined in:

- UNIX – *$SYBASE/$SYBASE_OCS/config/objectid.dat*

- NT – *%SYBASE%\%SYBASE_OCS\ini\objectid.dat*

In these examples, the SSL security service is specified on port number 2748(0x0abc).

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.

The sp_ssladmin is used to:

- Add local server certificates. You can add certificates and specify the password used to encrypt private keys, or require input of the password at the command line during start-up.
- Delete local server certificates.
- List server certificates.

The syntax for sp_ssladmin is:

```
sp_ssladmin {[addcert, certificate_path [, password|NULL]]
      [dropcert, certificate_path]
      [lscert]
      [help]}
```

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, 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, maximum 256 characters.
- Linux – 128 characters.
- IBM – both 32- and 64-bit platforms, 32 characters.
- HP – both 32- and 64-bit platforms, 8 characters.
- Digital UNIX – 80 characters.
- Windows NT – 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 log in.

When NULL is the password, you must start dataserver with a -y flag, which prompts the administrator for the private-key password at the command line.

After restarting Adaptive Server with an SSL connection established, use 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 isql. You can specify “localhost” as the *hostname* in the *interfaces* file (*sql.ini* on Windows) to prevent clients from connecting remotely. Only a local connection can be established, and the password is never transmitted over a network connection.

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. Typically, the additional I/O accrued during the SSL handshake may make user login 10 to 20 times slower. Also, SSL-enabled connections require more memory. You must have approximately 40K more memory for each user connection.

CipherSuites

During the SSL handshake, the client and server negotiate a common security protocol via a CipherSuite. **CipherSuites** are preferential lists of key-exchange algorithms, hashing methods, and encryption methods used by SSL-enabled applications. For a complete description of CipherSuites, visit the Internet Engineering Task Force (IETF) organization at http://www.ietf.org/rfc/rfc2246.txt.

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 CipherSuites that are available with the SSL Plus library API and the cryptographic engine, Security Builder™, both from Certicom Corp.

The following lists the CipherSuites, ordered by strength from stronget to weakest, supported in Adaptive Server 12.5and later.

```
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,
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
```

Note The CipherSuites listed above 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 CipherSuites.

Network-based security

Adaptive Server provides network-based security services that enable you to authenticate users and protect data transmitted among machines on a network.

In a distributed client/server computing environment intruders can view or tamper with confidential data. With Adaptive Server, you can use security services provided by third-party providers to authenticate users, encrypt data, and prevent data tampering.

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

- Unified login – use a security mechanism to authenticate users *once* without requiring them to supply a name and password every time they log in to an Adaptive Server.
- Message confidentiality – encrypt data over the network.
- Mutual authentication – use the security mechanism to verify the identity of the client and the server. (This must be requested by the client and cannot be required by Adaptive Server.)
- Message integrity – verify that data communications have not been modified.
- Replay detection – verify that data has not been intercepted by an intruder.
- Out-of-sequence check – verify the order of data communications.
- Message origin checks – verify the origin of the message.

- Remote procedure security – establish mutual authentication, message confidentiality, and message integrity for remote procedure communications.

Note The security mechanism you are using may not support all of these services.

Auditing

Adaptive Server includes a comprehensive audit system. The audit system consists of a system database called sybsecurity, configuration parameters for managing auditing, a system procedure, sp_audit, to set all auditing options, and a system procedure, sp_addauditrecord, to add user-defined records to the audit trail. When you install auditing, you can specify the number of audit tables that Adaptive Server will use 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, 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
- Logins and logouts

Auditing functionality is discussed in Chapter 11, “Auditing.”

User-defined login security

UDLS gives you more control over security-related features of Adaptive Server.

In Adaptive Server versions 12.0 and later, the System Security Officer can:

- Add more user logins and roles than was possible in earlier versions
- 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
- Lock and unlock roles manually
- Ensure that all user passwords have at least one digit
- Specify the minimum password length required server-wide or for a specific login or role
- Display all security-related information for logins and roles
- Associate a password expiration value with a specified login or role

Negative values may be used 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).

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.

v **Setting the server-wide *maximum failed logins***

- To set the server-wide maximum failed logins for logins and roles, use the maximum failed logins configuration parameter.

This example sets the system-wide maximum failed logins to 5:

```
sp_configure "maximum failed logins", 5
```

For details on the syntax and rules for using maximum failed logins, see sp_configure.

v Setting the *maximum failed logins* for specific logins

- To set the maximum failed logins for a specific login at creation, use sp_addlogin.

  This example creates the new login "joe" with the password "Djdiek3" and sets the maximum number of failed login attempts for the login "joe" to 2:

  ```
  sp_addlogin joe, "Djdiek3", pubs2, null, null, null,
  null, 2
  ```

  For details on the syntax and rules for using maximum failed logins, see sp_addlogin.

v 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 the intern_role role with the password "temp244", and sets the maximum failed logins for intern_role to 20:

  ```
  create role intern_role with passwd "temp244",
  maximum failed logins 20
  ```

  For details on the syntax and rules for using maximum failed logins, see create role.

v Changing the *maximum failed logins* for specific logins

- Use sp_modifylogin to set or change the maximum failed logins for an existing login.

  Example 1 Changes the maximum failed logins for the login "joe" to 40:

  ```
  sp_modifylogin "joe", @option="maximum failed
  logins", @value="40"
  ```

  Note The *value* parameter is a character datatype; therefore, quotes are required for numeric values.

  Example 2 Changes the overrides for maximum failed logins for all logins to 3:

```
sp_modifylogin "all overrides", @option="maximum
failed logins", @value="-1"
```

Example 3 Removes the overrides for maximum failed logins option for all logins:

```
sp_modifylogin "all overrides", "maximum failed
logins", "3"
```

sp_modifylogin only effects user roles, not system roles. For details on the syntax and rules for using maximum failed logins, see sp_modifylogin.

v **Changing the *maximum failed logins* for specific roles**

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

  Example 1 Changes the maximum failed logins allowed for physician_role to 5:

  ```
  alter role "all overrides" set maximum failed logins
  -1
  ```

  Example 2 Removes the overrides for the maximum failed logins for all roles:

  ```
  alter role physician_role set maximum failed logins
  5
  ```

  For details on the syntax and rules for using maximum failed logins, see alter role.

Locking and unlocking logins and roles

A login or role can be locked when:

- Its password expires, or
- The maximum number of failed login attempts occur, or
- The System Security Officer locks the login or role manually.

v **Locking and unlocking logins**

- The System Security Officer can use sp_locklogin to lock or unlock a login manually. (This is not new functionality, but is mentioned here for comparison to the new methods available for locking and unlocking roles.)

  For example:

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

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

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

For details on the syntax and rules for using sp_locklogin, see sp_locklogin.

v **Locking and unlocking roles**

- The System Security Officer can use alter role to lock or unlock a role manually.

  For example:

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

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

  For details on the syntax and rules for using lock and unlock, see alter role.

v **Unlocking logins and roles at server start-up**

- Automatic login lockouts can cause a site to end up in a situation in which all accounts capable of unlocking logins (System Administrators and System Security Officers) are locked. In these situations, use the -u flag with the dataserver utility to unlock a specific login or role when you start Adaptive Server.

  For details on the syntax and rules for using the -u flag, see the *Utility Guide*.

Displaying password information

This section discusses displaying password information for logins and roles.

v **Displaying password information for specific logins**

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

  This example displays information about the login joe:

  ```
  sp_displaylogin joe
  ```

  ```
  Suid: 2
  Loginame: joe
  Fullname: Joseph Resu
  Default Database: master
  Default Language:
  ```

```
Configured Authorization: intern_role (default OFF)
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
```

For details on the syntax and rules, see sp_displaylogin.

v **Displaying password information for specific roles**

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

  This example displays information about the physician_role role:

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

  For details on the syntax and rules, see sp_displayroles.

Checking passwords for at least one digit

The System Security Officer can tell the server to check for at least one character or 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:

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

This deactivates the check password functionality:

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

For details on the syntax and rules for using the new parameter, see sp_configure.

Setting and changing *minimum password length*

In previous versions, the minimum password length was a non-configurable, hard-coded value of six characters. 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. It does not affect existing passwords.

Setting the server-wide *minimum password length*

- Use the minimum password length configuration parameter to specify a server-wide value for minimum password length for both logins and roles.

  This example sets the minimum password length for all logins and roles to 7 characters:

  ```
  sp_configure "minimum password length", 7
  ```

  For details on the syntax and rules for using minimum password length, see sp_configure.

Setting *minimum password length* for a specific login

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

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

  ```
  sp_addlogin joe, "Djdiek3", @minpwdlen=8
  ```

  For details on the syntax and rules for using minimum password length, see sp_addlogin.

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.

For details on the syntax and rules for using minimum password length, see create role.

v **Changing *minimum password length* for a specific login**

- Use sp_modifylogin to set or change minimum password length for an existing login. sp_modifylogin only effects user roles, not system roles.

  Example 1 Changes minimum password length for the login "joe" to 8 characters.

  ```
  sp_modifylogin "joe", @option="min passwd length",
  @value="8"
  ```

  Note The *value* parameter is a character datatype; therefore, quotes are required for numeric values.

  Example 2 Changes the value of the overrides for minimum password length for all logins to eight characters.

  ```
  sp_modifylogin "all overrides", @option="min passwd
  length", @value="8"
  ```

  Example 3 Removes the overrides for the minimum password length for all logins.

  ```
  sp_modifylogin "all overrides", "min passwd length",
  @value="-2"
  ```

  For details on the syntax and rules for using minimum password length, see sp_modifylogin.

v **Changing *minimum password length* for a specific role**

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

  Example 1 Sets the minimum length for physician_role, an existing role, to 5 characters:

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

  Example 2 Overrides the minimum password length for all roles:

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

For details on the syntax and rules for using minimum password length, see alter role.

Setting the expiration interval for a password

System Administrators and System Security Officers can:

Use	To
sp_addlogin	Specify the expiration interval for a login password at creation
sp_modifylogin	Change the expiration interval for a login password. sp_modifylogin only effects user roles, not system roles.
create role	Specify the expiration interval for a role password at creation
alter role	Change the expiration interval for a role password

The following rules apply to password expiration for logins and roles:

- A password expiration interval assigned to individual login accounts or roles overrides the global password expiration value. This allows you to specify shorter expiration intervals for sensitive accounts or roles, such as System Security Officer passwords, and more relaxed intervals for less sensitive accounts such as an anonymous login.
- A login or role for which the password has expired is not directly activated.

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

Password expiration turned off for pre-12.x passwords

Password expiration did not affect roles in versions prior to Adaptive Server 12.x. Therefore, in Adaptive Server 12.x and later, password expiration is deactivated for any existing user-defined role passwords. During the upgrade all user-defined role passwords are stamped as having a password interval of 0.

Message for impending password expiration

When a password for a login or role is about to expire, a warning message asks the user to contact the System Security Officer.

Circumventing password protection

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

If a System Security Officer wants to bypass the password mechanism for certain users, the System Security Officer can grant the password-protected role to another role and grant this new 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 the fictitious company 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 now has the ability to 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 sp_addlogin 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:

```
sp_addlogin joe, "Djdiek3", null, null, null, 2
```

For details on the syntax and rules for using the new parameter, see sp_addlogin.

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
```

For details on the syntax and rules for using passwd expiration, see create role.

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 sp_modifylogin 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. sp_modifylogin only effects user roles, not system roles.

Example 1 Changes the password expiration interval for the login “joe” to 5 days:

```
sp_modifylogin "joe", @option="passwd expiration", @value="5"
```

Note The *value* parameter is a character datatype; therefore, quotes are required for numeric values.

Example 2 Changes the value of the overrides for the password expiration for all logins to 3 days”:

```
sp_modifylogin "all overrides", @option="passwd expiration", @value="3"
```

Example 3 Removes the value of the overrides for the password expiration for all logins:

```
sp_modifylogin "all overrides", @option="passwd expiration", @value="-1"
```

For details on the syntax and rules for using passwd expiration, see sp_modifylogin.

CHAPTER 10

Managing Adaptive Server Logins, Database Users, and Client Connections

This chapter describes methods for managing Adaptive Server login accounts and database users.

Adding new users: An overview

The process of adding new logins to Adaptive Server, adding users to databases, and granting them **permission** to use commands and database objects is divided among the System Security Officer, System Administrator, and Database Owner.

Note The "Adding new users" procedure creates login accounts for a particular server using sp_addlogin, which stores account information in the *syslogins* table on that server. You can also create and store login accounts on a LDAP server. See "Creating and managing Adaptive Server logins using LDAP" on page 386.

---

v **Adding new users**

1. A System Security Officer uses sp_addlogin to create a server login account for a new user.
2. A System Administrator or Database Owner uses sp_adduser to add a user to a database. This command can also give the user an alias or assign the user to a group. For more information, see "Creating groups" on page 348.
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. Users or groups can also be granted permission to grant specific permissions on objects to other users or groups. See Chapter 12, "Managing User Permissions" for detailed information about permissions.

Table 10-1 summarizes the system procedures and commands used for these tasks.

Table 10-1: Adding users to Adaptive Server and databases

Task	Required role	Command or procedure	Database
Create new logins, assign passwords, default databases, default language, and full name	System Security Officer	sp_addlogin	Any database
Create groups	Database Owner or System Administrator	sp_addgroup	User database
Create and assign roles	System Security Officer	create role	

Task	Required role	Command or procedure	Database
Add users to database, assign aliases, and assign groups	Database Owner or System Administrator	sp_adduser	User database
Grant groups, users, or roles permission to create or access database objects	Database Owner, System Administrator, or object owner	grant	User database

Choosing and creating a password

Your password helps prevent access by unauthorized people. When you create your password, follow these guidelines:

- 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.

Follow these rules to create a password:

- Passwords must be at least 6 bytes long.
- Passwords can consist of any printable letters, numbers, or symbols.
- A password must be enclosed in quotation marks in sp_addlogin if it:
  - Includes any character other than A–Z, a–z, 0–9,_, #, valid single-byte or multibyte alphabetic characters, or accented alphabetic characters
  - Begins with a number 0–9

Adding logins to Adaptive Server

Use sp_addlogin to add a new **login** name to Adaptive Server. You do not use it to give the user permission to access user databases. Use sp_adduser for that purpose. Only the System Security Officer can execute sp_addlogin. The syntax is:

```
sp_addlogin loginname, passwd [, defdb]
    [, deflanguage [, fullname]]]
```

where:

- *loginame* – is the new user's login name. The login name must follow the rules for identifiers and must be unique on Adaptive Server. To simplify both the login process and server administration, make the Adaptive Server login name the same as the user's operating system login name. This makes logging in to Adaptive Server easier because many client programs use the operating system login name as a default. It also simplifies management of server and operating system login accounts, and makes it easier to correlate usage and audit data generated by Adaptive Server and by the operating system.
- *passwd* – is the password for the new user. For guidelines on choosing and creating secure passwords, see "Choosing and creating a password" on page 345. For information on changing a password, see "Changing passwords" on page 368.
- *defdb* – is the **default database**, where the user starts each session of Adaptive Server.

  Note The default database is master. To discourage users from creating database objects in the master database, assign a default database other than master to most users.

  A System Administrator can change anyone's default database with sp_modifylogin. Other users can change only their own default database.

  After specifying the default database, add the user to the default database with sp_adduser so that he or she can log in directly to the default database.
- *deflanguage* – is the **default language** in which the user's prompts and messages are displayed. If you omit this parameter, Adaptive Server's default language is used. A System Administrator can change any user's default language with sp_modifylogin. Other users can change only their own language.

- *fullname* – is the full name of the user. This is useful for documentation and identification purposes. If omitted, no full name is added. A System Administrator can change any user's full name with sp_modifylogin. Other users can change only their own full name.

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

```
sp_addlogin "maryd", "100cents"
```

The password requires quotation marks because it begins with 1.

After this statement is executed, "maryd" can log into 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") for user and makes pubs2 the default database for this user:

```
sp_addlogin omar_khayyam, rubaiyat, pubs2
```

To specify a full name for a user and use the default database and language, you must specify null in place of the *defdb* and *deflanguage* parameters. For example:

```
sp_addlogin omar, rubaiyat, null, null,
    "Omar Khayyam"
```

Alternatively, you can specify a parameter name, in which case you do not have to specify all the parameters. For example:

```
sp_addlogin omar, rubaiyat,
    @fullname = "Omar Khayyam"
```

When you execute sp_addlogin, Adaptive Server adds a row to master.dbo.syslogins, assigns a unique server **user ID** (suid) for the new user, and fills in other 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 human-readable.

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 sp_addlogin is executed.

Login failure to Adaptive Server

Adaptive Server must successfully authenticate a user before they are able to access data in Adaptive Server. If the authentication attempt fails, Adaptive Server returns the following message and the network connection is terminated:

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

This message is a generic login failure message that does not tell the connecting user whether the failure resulted from a bad user name or a bad password. This generic message guards against malicious attempts to gain access to Adaptive Server.

Creating groups

Groups provide a convenient way to grant and revoke permissions to more than one user in a single statement. Groups enable 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. Every user is a member of the group “public” and can also be a member of one other group. (Users remain in “public,” even when they belong to another group.)

It is probably most convenient to create groups before adding users to a database, since sp_adduser can assign users to groups as well as add them to the database.

A System Administrator or the Database Owner can create a group at any time with sp_addgroup. The syntax is:

sp_addgroup *grpname*

The group name, a required parameter, must follow the rules for identifiers. The System Administrator can assign or reassign users to groups with sp_changegroup.

To set up the Senior Engineering group, use the following 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.

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.

  You can use this feature 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.

  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 372, 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 370 for information about modifying a user's group membership.

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 sp_addlogin
- Can use sp_modifylogin to make that database the default

This example shows how a Database Owner could 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 12, “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 issues the use *database_name* command, and his or her name is not found in the database’s sysusers or sysalternates table, 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 add a guest entry to the sysusers table of the database with sp_adduser:

```
sp_adduser guest
```

The guest user can be removed with sp_dropuser, as discussed in “Dropping users” on page 364.

If you drop the guest user from the master database, server users who have not yet been added to any databases will be unable to log in to Adaptive Server.

Note Although more than one individual can be a guest user in a database, you can still use the user's server user ID, which is unique within the server, to audit each user's activity. For more information about auditing, see Chapter 11, "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 12, "Managing User Permissions," for a description of the "public" privileges.

When you install Adaptive Server, master..sysusers contains a guest entry. The installation script for the pubs2 database also contains a guest entry for its sysusers table.

"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. This saves the owner from having to use sp_adduser to explicitly name each one 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 could grant select permission on titles to all database users except "guest" by executing these commands:

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

"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 User's 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

Creating visitor accounts

The System Security Officer can use sp_addlogin 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. Setting up a visitor account to be used by more than one user is not recommended because you lose individual accountability.

You can add a visitor user account named “guest” to master..syslogins using sp_addlogin. This “guest” user account takes precedence over the system “guest” user account. Note that, 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, both the local and the remote server must be configured. For information about setting up remote servers and adding remote users, see Chapter 13, “Managing Remote Servers.”

Note Remote users and remote procedure calls are not included in the evaluated configuration.

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

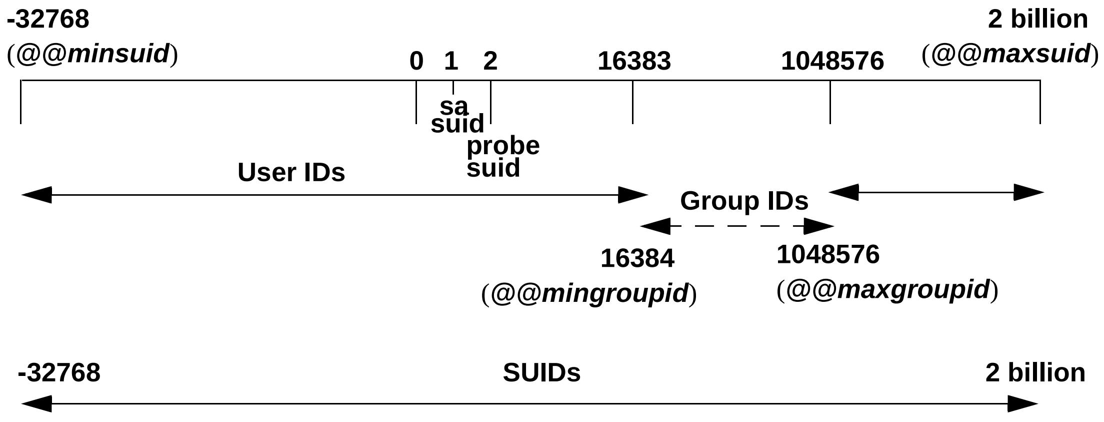
Table 10-2 describes the valid ranges for the ID types.

Table 10-2: Ranges for ID types

ID type	Server limits
Logins per server (*suid*)	2 billion plus 32K
Users per database (*uid*)	2 billion less 1032193
Groups per database (*guid*)	16,390 to 1,048,576

Figure 10-1 illustrates the limits and ranges for logins, users, and groups.

Figure 10-1: Users, groups, and logins available in Adaptive Server



Although Adaptive Server can handle over 2 billion users connecting at one time, the actual number of users that can connect to Adaptive Server is limited by:

- The number of user connections configuration parameter
- The number of file descriptors available from the operating system. Each user login uses one file descriptor per connection

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

See the release bulletin for the most up to date information about Adaptive Server's limits for logins, users, and groups.

Login connection limitations

Although Adaptive Server allows you to define over 2,000,000,000 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
- The number of file descriptors available for Adaptive Server. Each login uses one file descriptor for the connection.

Note The datatype of the server process ID (*spid*) has not been changed. Therefore, the maximum number of concurrent tasks running on the server is still 32,000.

---

v **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.

Viewing server limits for logins, users, and groups

Table 10-3 lists the global variables for the server limits of logins, users, and groups:

Table 10-3: Global variables for logins, users, and groups

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

To issue a global variable, enter:

```
select variable_name
```

For example:

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

Creating and assigning roles to users

The final steps in adding database users are assigning them special roles, as required, and granting permissions. For more information on permissions, see Chapter 12, “Managing User Permissions.”

The roles supported by Adaptive Server enable you to enforce 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. Object owners can grant database access as appropriate to each role.

Table 10-4 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.

Table 10-4: System roles and related tasks

Role	**Value for *role_granted***	**Description**
System Administrator	sa_role	Manages and maintains Adaptive Server databases and disk storage
System Security Officer	sso_role	Performs security-related tasks
Operator	oper_role	Backs up and loads databases server-wide

Note The sybase_ts_role, replication_role, and navigation_role roles are not included in the evaluated configuration.

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 will be mutually exclusive
- Whether such exclusivity will be at the membership level or activation level

User-defined role names cannot duplicate user names.

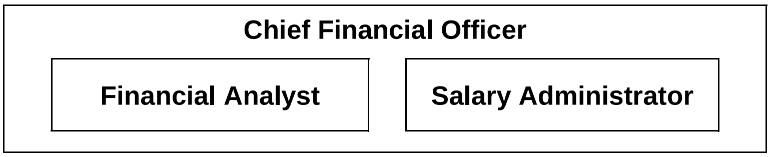
Avoid name-conflicts when you create user-defined roles by following a naming convention. For example, you could use the "_role" suffix for role names. Adaptive Server does not check for such restrictions.

If a role must have the same name as a user, you can avoid conflict by creating a new role, having it contain the original role, and then granting the new role to the user.

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. For example, the "chief_financial_officer" role might contain both the "financial_analyst" and the "salary_administrator" roles, as shown in Figure 10-2.

Figure 10-2: Role hierarchy



The Chief Financial Officer can perform all tasks and see all data that can be viewed by the salary administrators and financial analysts.

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 “tech_support” roles. You might also 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.

Configuring user-defined roles

After you have planned the roles to create and the relationships among them, configure your system for user-defined roles with the max roles enabled per user configuration parameter.

The maximum number of roles that a user can activate per user session is 127. The default value is 20. The minimum number of roles, which is 10, includes the system roles that come with Adaptive Server.

The maximum number of roles that can be activated server-wide is 992. The first 32 roles are reserved for Sybase system roles.

Creating a user-defined role

Use the create role command to create a role. The syntax is:

create role *role_name* [with passwd "*password*"]

where:

- *role_name* – is the name of a new role.
- *password* – is an optional password that must be specified by the user who will use the role.

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"
```

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. The syntax is:

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
```

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
```

To define sso_role and sa_role as mutually exclusive at the activation level, enter:

```
alter role sso_role add exclusive activation sa_role
```

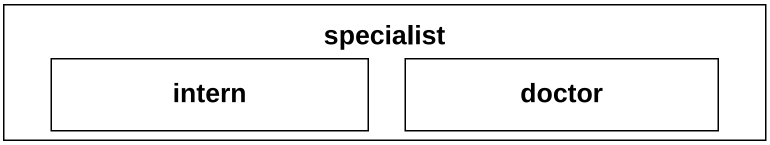
Defining and changing a role hierarchy

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
```

Figure 10-3: Creating a role hierarchy



In Figure 10-3, the “specialist” role contains the “doctor” and “intern” roles. This means that “specialist” has all the privileges of both “doctor” and “intern.”

To establish a hierarchy with a “super_user” role containing the sa_role and oper_role system roles, specify:

```
grant role sa_role to super_user
grant role oper_role to super_user
```

Note If a role that requires a password is contained within another role, the user with the role that contains the other does not need to use the password for the contained role. For example, in Figure 10-3, say 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.

  For example, in Figure 10-3, you cannot grant “doctor” to “specialist” because “specialist” already contains “doctor.”

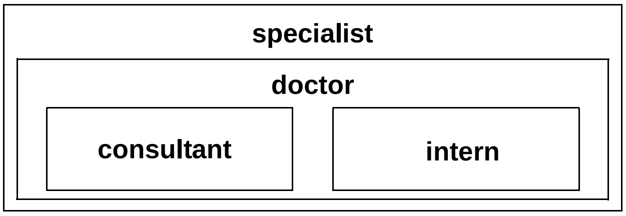
- You can grant a role to another role that does not directly contain it.

  For example, in Figure 10-4, 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” would still contains “intern.”

  In Figure 10-4, “doctor” has “consultant” role permissions because “consultant” has been granted “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.

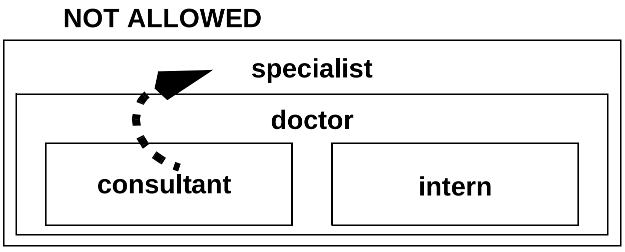
Figure 10-4: Explicitly and implicitly granted privileges



- 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 10-5, you cannot grant the “specialist” role to the “consultant” role; “consultant” is already contained in “specialist”.

Figure 10-5: Granting a role to a role contained by grantor

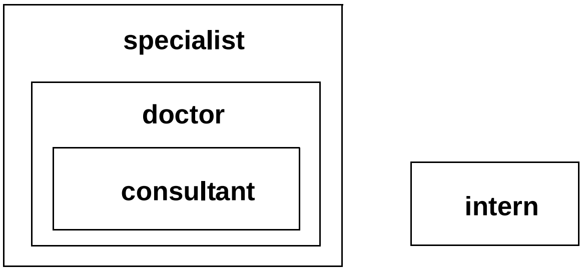


- When the System Security Officer grants a user a role that contains other roles, the user implicitly gets membership in all roles contained by the granted role. However, a role can only be activated or deactivated directly if the user has explicit membership in that role.

- 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 10-6, 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 10-6: Mutual exclusivity at membership

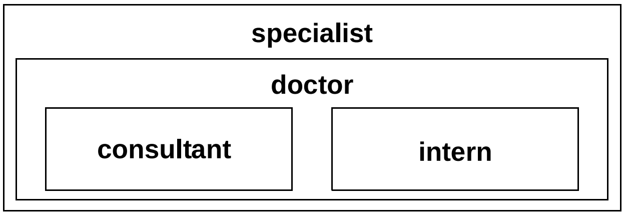


- The user can activate or deactivate only directly granted roles.

  For example, in the hierarchy shown in Figure 10-6, 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. For information, see "Activating and deactivating roles" on page 363.

  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, as shown in Figure 10-7:

Figure 10-7: Effect of revoking roles on role hierarchy



For example, in Figure 10-7:

- 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 a role's default setting for any user. Individual users can change only their own default settings.

When a user logs in to Adaptive Server, the user's roles are not necessarily active, depending upon the default that is set for the 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 or user determines whether to activate any roles granted by default at login. sp_modifylogin sets the default status of user roles individually for each user. sp_modifylogin only effects user roles, not system roles.

By default, user-defined roles are not activated at login, but system roles are automatically activated, if they do not have passwords associated with them.

To set up a role to activate at login:

```
sp_modifylogin loginname, "add default role", role_name
```

To ensure that a role is inactive at login:

```
sp_modifylogin loginname, "drop default role", role_name
```

For example, to change the default setting for Ralph's intern_role to be active automatically at login, execute:

```
sp_modifylogin ralph, "add default role", intern_role
```

Activating and deactivating roles

Roles must be active to have the access privileges of each role. Depending on the default set for a role, the role may or may not be active at login. If the role has a password, it will always be inactive at login.

To activate or deactivate a role:

```
set role role_name [on|off]
```

To activate or deactivate a role that has an attached password, use:

```
set role role_name with passwd "password" [on|off]
```

For example, to activate the “financial_analyst” role with the password “sailing19”, enter:

```
set role financial_analyst with passwd "sailing19" on
```

You should activate roles only when you need them and turn them off when you no longer need them. For example, when the sa_role is active, you assume the identity of Database Owner within any database that you use. To turn off the System Administrator role and assume your “real” user identity, use:

```
set role sa_role off
```

If you are granted a role during a session, and you want to activate it immediately, use set role to turn it on.

Dropping users, groups, and user-defined roles

Table 10-5 list the system procedures that allow a System Administrator or Database Owner to drop users and groups.

Table 10-5: Dropping users and groups

Task	Required role	System procedure	Database
Drop user from database	Database Owner or System Administrator	sp_dropuser	User database
Drop group from database	Database Owner or System Administrator	sp_dropgroup	User database

Dropping users

A Database Owner 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:

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 with sp_dropuser. To deny access to a user who owns objects, use sp_locklogin to lock his or her account.

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

Dropping groups

Use 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_changegroup, discussed in "Changing a user's group membership" on page 370.

Dropping user-defined roles

To drop a role, use the following, where *role_name* is the name of a user-defined role:

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

with override revokes all access privileges granted to the role in every database server-wide.

If the role has any access privileges already granted, 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, or
- Use the with override option with the drop role command. The with override option ensures that Adaptive Server automatically removes permission information for the role from all databases.

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.

Locking or dropping Adaptive Server login accounts

To prevent a user from logging in to Adaptive Server, you can either lock or drop an Adaptive Server login account. Locking a login is safer than dropping it because locking a login account maintains the suid so that it cannot be reused.

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 sp_droplogin 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 belongs to the last remaining System Security Officer or System Administrator

Table 10-6: Locking or dropping login accounts

Task	Required role	System procedure	Database
Lock login account, which maintains the suid so that it cannot be reused	System Administrator or System Security Officer	sp_locklogin	master
Drop login account, which allows reuse of suid	System Administrator	sp_droplogin	master

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 Administrator or a System Security Officer to use sp_locklogin.

The syntax is:

```
sp_locklogin [loginame, "{lock | unlock}"]
```

where:

- *loginame* is the name of the account to be locked or unlocked. It must be an existing valid account.
- 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.

Dropping login accounts

A System Administrator can use sp_droplogin to deny a user access to Adaptive Server. The syntax is:

```
sp_droplogin loginame
```

You cannot use sp_droplogin to drop a user from any database on the server. Use sp_dropuser to drop the user from a database. You cannot drop a user from a database if that user owns any objects in the database. For more information, see "Dropping users" on page 364.

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 threshold cannot execute the stored procedure.
- Threshold procedures are executed with the most limited set of the roles assigned to the user. The user must have both of the following:
  - The set of roles active for the user at the time the threshold was added or last modified, and
  - The set of roles directly granted to the user at the time the threshold "fires."

- If a threshold requires a particular role, that role must be active for the user when the threshold is created. If that role is later revoked, the threshold cannot execute the 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.

Changing user information

Table 10-7 lists the system procedures you use to change passwords, default database, default language, full name, or group assignment.

Table 10-7: System procedures for changing user information

Task	Required role	System procedure	Database
Change your password	None	sp_password	Any database
Change another user's password	System Security Officer	sp_password	Any database
Change your default database, default language, or full name	None	sp_modifylogin	Any database
Change a login account's default database, default language, or full name	System Administrator	sp_modifylogin	Any database
Change the group assignment of a user	System Administrator, Database Owner	sp_changegroup	User database

Changing passwords

All users can change their passwords at any time using sp_password. The System Security Officer can use sp_password to change any user's password. The syntax is:

sp_password *caller_passwd*, *new_passwd* [, *loginame*]

where:

- *caller_passwd* is the password of the login account that is currently executing sp_password.
- *new_passwd* is the new password for the user executing sp_password, or for the user indicated by *loginame*. For guidelines on choosing and creating secure passwords, see "Choosing and creating a password" on page 345.

- *loginame* can be used only by a System Security Officer to change another user's password.

For example, a user can change his or her own password from "3blindmice" to "2mediumhot" using:

```
sp_password "3blindmice", "2mediumhot"
```

These passwords are enclosed in quotes because they begin with numbers.

In the following example, the System Security Officer whose password is "2tomato" changes Victoria's password to "sesame1":

```
sp_password "2tomato", sesame1, victoria
```

Requiring new passwords

Your site 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. For information, see Chapter 4, "Setting Configuration Parameters." Even if you do not use systemwide password expiration, it is important, for security reasons, that users change their passwords periodically.

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, 1997:

```
select name, pwdate
from syslogins
where pwdate < "Sep 15 1997"
```

Null passwords

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

```
sp_password null, "8M4LNCH"
```

Note "null" is not enclosed in quotes in the statement.

Changing user defaults

Any user can use sp_modifylogin to change his or her default database, default language, or full name. sp_modifylogin only effects user roles, not system roles. A System Administrator can change these settings for any user. The syntax is:

sp_modifylogin *account*, *column*, *value*

where:

- *account* – is the name of the user whose account you are modifying.
- *column* – specifies the option that you are changing. The options are:
  - defdb – the "home" database to which the user is connected when he or she logs in
  - deflanguage – the official name of the user's default language, as stored in master..syslanguages
  - fullname – the user's full name
- *value* – is the new value for the specified option.

After you execute sp_modifylogin to change the default database, the user is connected to the new default database the next time he or she logs in. However, sp_modifylogin does not automatically give the user access to the database. Unless the Database Owner has set up 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 "anna" to pubs2:

```
sp_modifylogin anna, defdb, pubs2
```

This example changes the default language for "claire" to French:

```
sp_modifylogin claire, deflanguage, french
```

This example changes the full name for "mtwain" to "Samuel Clemens."

```
sp_modifylogin mtwain, fullname, "Samuel Clemens"
```

Changing a user's group membership

A System Administrator 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. (Use sp_addgroup to create a group.)
- 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 "manage," use:

```
sp_changegroup manage, 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 that have been granted to the new group.

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

Changing the user process 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, and *application_name* is the application that is connecting to Adaptive Server. These parameters are stored in the clientname, clienthostname, 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 spid to view their connection information. For example, if the client "alison" described above connects with a spid of 13, issue the following command to view all the connection information for this client:

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

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

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

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. For more information about auditing, see Chapter 11, “Auditing.”

Table 10-8 lists the system procedures used to manage aliases:

Table 10-8: System procedures for managing aliases

Task	Require role	System procedure	Database
Add an alias for a user	Database Owner or System Administrator	sp_addalias	User database
Drop an alias	Database Owner or System Administrator	sp_dropalias	User database

Note As of version 12.0, you cannot drop the alias of a login if that login created objects in the database. In most cases, you should 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. The syntax is:

```
sp_addalias loginame, name_in_db
```

where:

- *loginame* – is the name of the user who wants an alias in the current database. This user must have an account 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 both master..syslogins and 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 real 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 for a user who owns objects in the database that were created with version 12.0 or later. You must drop the objects (re-creating them under a different login, if needed) before you can drop the alias.

Getting information about aliases

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

```
sp_helpuser dbo

Users_name      ID_in_db    Group_name   Login_name
```

```
----------     --------    ----------    ----------
dbo            1           public        sa

(1 row affected)

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

(4 rows affected)
```

Getting information about users

Table 10-9 lists procedures you can use to obtain information about users, groups, and current Adaptive Server usage.

Table 10-9: Reporting information about Adaptive Server users and groups

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

Getting reports 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 give 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 being run, sp_who reports the server process ID, its status, the login name of the process user, the name of the host computer, the server process ID of a process that's blocking this one (if any), the name of the database, and the command being run.

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

The following example shows the results of executing sp_who without a parameter:

```
spid   status   loginame hostname blk dbname cmd
------ -------- -------- -------- --- ------ ----------------
     1 running  sa       sunbird  0   pubs2  SELECT
     2 sleeping NULL              0   master NETWORK HANDLER
     3 sleeping NULL              0   master MIRROR HANDLER
     4 sleeping NULL              0   master AUDIT PROCESS
     5 sleeping NULL              0   master CHECKPOINT SLEEP

(5 rows affected, return status = 0)
```

sp_who reports NULL for the *loginame* for all system processes

Getting information about login accounts

Use sp_displaylogin to display information about a specified login account, including any roles granted to that account, where *loginame* is the user login account about which you want information:

sp_displaylogin [*loginame*]

If you are not a System Security Officer or System Administrator, you can get information only about your own account. If you are a System Security Officer or System Administrator, you can use the *loginame* 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, and whether your account is locked.

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 is displayed.

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:

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

Finding user names and IDs

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

Table 10-10: System functions suser_id and suser_name

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

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

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

```
select suser_id("sandy")

------
     3
```

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

```
select suser_name(), suser_id()
------------------------------ ------
mary                                4
```

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

Table 10-11: System functions user_id and user_name

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

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

```
select user_name(10)
select user_name( )
select user_id("joe")
```

Displaying information about roles

Table 10-12 lists the system procedures and functions to use to find information about roles and the section in this chapter that provides details.

Table 10-12: Finding information about roles

To display information about	Use	See
The role ID of a role name	role_id system function	“Finding role IDs and names” on page 379
The role name of a role ID	role_name system function	“Finding role IDs and names” on page 379
System roles	show_role system function	“Viewing active roles” on page 379
Role hierarchies and roles that have been granted to a user or users	sp_displayroles system procedure	“Displaying a role hierarchy” on page 380
Whether one role contains another role in a role hierarchy	role_contain system function	“Viewing user roles in a hierarchy” on page 380
Whether two roles are mutually exclusive	mut_excl_roles system function	“Determining mutual exclusivity” on page 380
Roles that are active for the current session	sp_activeroles system procedure	“Determining role activation” on page 380

To display information about	Use	See
Whether you have activated the correct role to execute a procedure	proc_role system function	"Checking for roles in stored procedures" on page 381
Logins, including roles that have been granted	sp_displaylogin system procedure	"Getting information about login accounts" on page 376
Permissions for a user, group, or role	sp_helprotect system procedure	"Reporting on permissions" on page 487

Finding role IDs and names

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

```
role_id(role_name)
```

Any user can execute role_id. If the role is valid, role_id returns the server-wide ID of the role (srid). The syssrvroles system table contains an srid column with the role ID and a name column with the role name. If the role is not valid, 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 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 give information about user-defined roles.

Displaying a role hierarchy

You can see all roles granted to your login name or see the entire hierarchy tree of roles displayed in table format using sp_displayroles:

```
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 or the System Administrator 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* contains *role2*, role_contain returns 1.

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(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:

```
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 proc_role within a stored procedure to guarantee that only users with a specific role can execute the procedure. Only proc_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. If you want to ensure, for example, that all users who are not System Administrators can never be granted permission to execute a stored procedure, use proc_role within the stored procedure itself. It checks to see whether the invoking user has the correct role to execute the procedure.

proc_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 proc_role to see if the user has the sa_role role:

```
create proc test_proc
as
if (proc_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
```

Monitoring license use

The License Use Monitor allows a System Administrator to monitor the number of user licenses used in Adaptive Server and 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 License Use Monitor is turned off when Adaptive Server is first installed or upgraded. The System Administrator must configure the License Use Monitor to monitor license usage. See "Configuring License Manager to monitor user licenses" on page 382 for configuration information.

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, it counts as one license. However, if the user logs in once from host A, and once from host B, it counts as two licenses. 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 is counted.

Configuring License Manager to monitor user licenses

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.

The configuration parameter housekeeper free write percent must be set to 1 or more in order for the License Manager to track license use.

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. When the housekeeper task checks licenses, 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 message to the error log:

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

The housekeeper chores task runs during Adaptive Server's idle cycles. The housekeeper monitors the number of user licenses only if the license information configuration parameter is set to 1 or greater.

For more information about the housekeeper chores task, see Chapter 4, Using ENgines and CPUs," in *Performance and Tuning: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 10-13.

Table 10-13: Columns in syblicenseslog table

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

syblicenseslog looks similar to this:

```
status logdate                    maxlicenses
------ -------------------------- -----------
     0        Jul 17 1998 11:43AM         123
     0        Jul 18 1998 11:47AM         147
     1        Jul 19 1998 11:51AM         154
     0        Jul 20 1998 11:55AM         142
       0        Jul 21 1998 11:58AM           138
       0        Jul 21 1998  3:14PM           133
```

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

If Adaptive Server is shut down, License Manager 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.

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 kept 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 you want 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, the System Administrator would run sp_clearstats once a month.

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

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.

Creating and managing Adaptive Server logins using LDAP

LDAP is an industry standard for accessing directory services over a network. You can store server information on a centralized LDAP server rather than in an interfaces file—see “LDAP as a directory service” on page 16.

You can also use LDAP to store and manage user access to Adaptive Server.

With LDAP services enabled:

- Adaptive Server authenticates clients with data from an LDAP server.

  Users authenticate with passwords stored on an LDAP server rather than in the syslogins catalog. The LDAP server provides a centralized location for login accounts—both names and passwords.

- Adaptive Server servers share user login data stored on the LDAP server.

  Information formerly stored in syslogins is now managed and stored on an LDAP server. It is cached locally to preserve referential integrity and for other, database-specific uses.

  Note Adaptive Server group, role, and database user information continues to be stored and managed on Adaptive Server, not on the LDAP server.

- With LDAP enabled, users have a single login and password throughout the enterprise.

- Single sign-on is available with Kerberos support. If Kerberos is enabled, user authentication is handled using Kerberos.

Setup

Adaptive Server support for LDAP requires an ASE_DIRS license.

v **Setting up LDAP user accounts for Adaptive Server**

1. Add user login accounts to the LDAP server using a third-party LDAP tool such as the SunONE Directory Server, the Microsoft Active Directory, the Novell Directory Server, or the IBM SecureWay Directory Server.
2. Construct an LDAP URL search string for user authentication to Adaptive Server using sp_ldapadmin. For example:

   ```
   sp_ldapadmin set_primary_url,
       'ldap://voyager:389/ou=People,
       dc=MyCompany,dc=com??sub?uid=*'
   ```

   Your LDAP server determines the syntax and attribute support of the search string. For syntax and usage information for sp_ldapadmin, see *Reference Manual: Procedures*.
3. Enable LDAP user support using the Adaptive Server configuration parameter enable ldap user auth. For example:

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

   See “enable ldap user auth” on page 194 for syntax and usage information.

Migration

To aid migration of user accounts to the LDAP server, the configuration parameter enable ldap user auth provides a setting that allows authentication through either the LDAP server or syslogins.

When enable ldap user auth is 1, Adaptive Server first searches the LDAP server to authenticate a user. If that search fails, Adaptive Server searches syslogins. Thus, all users have access to Adaptive Server during the migration process—whether their accounts are currently on LDAP server or syslogins.

See “enable ldap user auth” on page 194 for syntax and usage information..

Managing user accounts locally

After a user account is added to the LDAP server, Adaptive Server can modify local characteristics of that account. For example, Adaptive Server creates a new row in syslogins for a user account when the LDAP server successfully authenticates the user and the user logs in to Adaptive Server.

Table 10-14 describes changes in syslogins with each login attempt.

Table 10-14: Changes in syslogins

A row already exists in syslogins for the user	LDAP authentication succeeds	Resulting change in syslogins
no	yes	Adds a new row for the user
no	no	No change
yes	yes	Updates an existing row if a new password is used
yes	no	No change

A System Administrator or a System Security Officer can add a row in syslogins using sp_addlogin to set login-specific values—such as a default database or the granting of roles—even before the user first logs in to Adaptive Server via LDAP. However, for sp_addlogin to succeed, the value of enable ldap user auth must allow LDAP authentication (either 1 or 2) and the user must have an LDAP account.

Note If LDAP authentication is enabled, you cannot change the value of the password used to authenticate the user at login using sp_addlogin. The password for LDAP authentication is always stored and managed at the LDAP server, and can be modified with LDAP server tools.

When a user login account is deleted from the LDAP server, the user account remains on Adaptive Server. The System Administrator or System Security Officer can delete the account only after all objects and users for that user account are deleted.

Failover support

Sybase provides failover support for the LDAP server. You can set up two LDAP URL search strings—one for the primary LDAP server and one for the secondary LDAP server—using sp_ldapadmin. If both URL search strings have been created, Adaptive Server uses the primary search string as long as that search string is valid and the primary LDAP server is active. Should the primary fail to respond, Adaptive Server uses the secondary URL search string.

CHAPTER 11 Auditing

This chapter describes how to set up auditing for your installation.

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 logins and logouts, server boots, 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 have 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
- Logins and logouts

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 have to 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

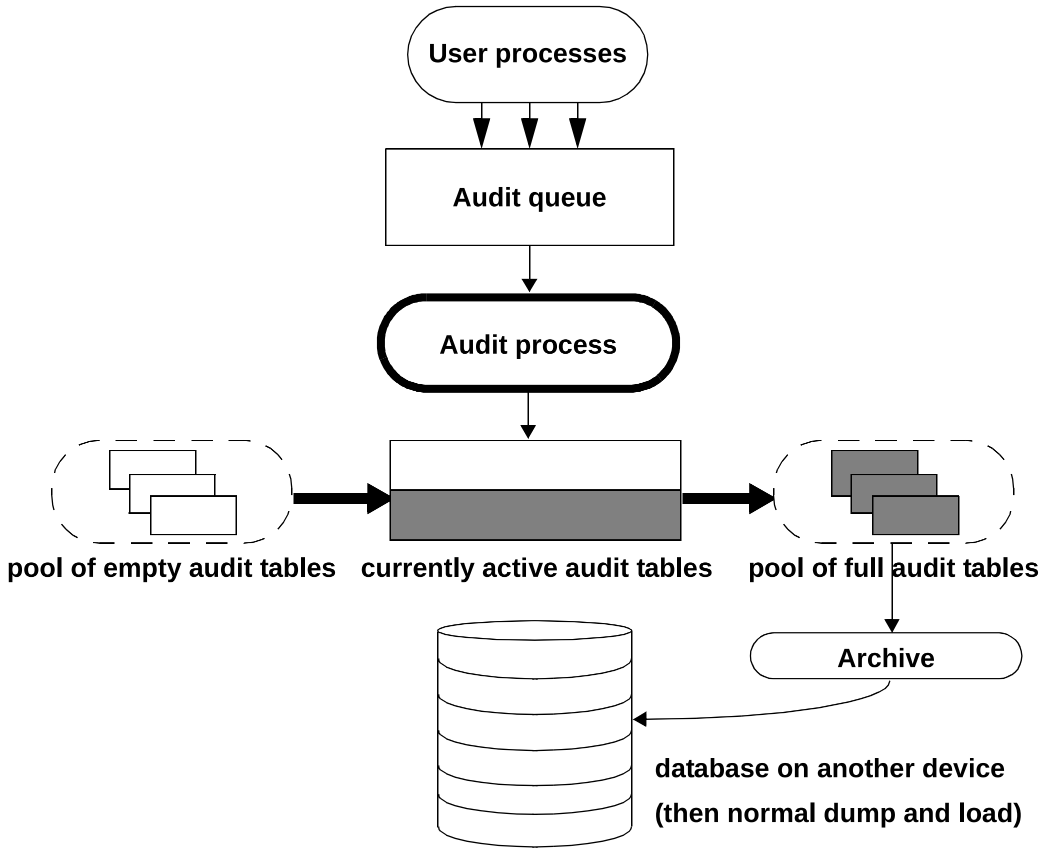
Adaptive Server stores the audit trail in system tables named sysaudits_01 through sysaudits_08. When you install auditing, you determine the number of audit tables for your installation. For example, if you choose to have two audit tables, they are named sysaudits_01 and sysaudits_02. At any given time, 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.

The recommended number of tables is two or more 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, refer to "Single-table auditing" on page 408 for instructions.

Figure 11-1 shows how the auditing process works with multiple audit tables.

Figure 11-1: Auditing with multiple audit tables


User processes
Audit queue
Audit process
pool of empty audit tables
currently active audit tables
pool of full audit tables
Archive
database on another device
(then normal dump and load)

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. For more information about managing the audit trail, see "Setting up audit trail management" on page 400.

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 whole 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.

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.

Installing and setting up auditing

Table 11-1 provides a general procedure for setting up auditing.

Table 11-1: General procedure for auditing

Action	Description	See
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.	“Installing the audit system” on page 396 and the Adaptive Server installation and configuration documentation
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.	“Setting up audit trail management” on page 400 For single-table auditing, “Single-table auditing” on page 408
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.	“Setting up transaction log management” on page 406
4. Set auditing options.	Using sp_audit to establish the events to be audited.	“Setting global auditing options” on page 412
5. Enable auditing.	Using sp_configure to turn on the auditing configuration parameter. Adaptive Server begins writing audit records to the current audit table.	“Enabling and disabling auditing” on page 407

Installing the audit system

The audit system is usually installed with auditinit, the Sybase installation program. Alternatively, you can install auditing without auditinit. For details, see “Installing auditing with installsecurity” on page 397. Installation and auditinit are discussed in the Adaptive Server installation and configuration documentation for your platform.

When you install auditing, you can establish the number of system tables you want to use for the audit trail, the device for each audit system table, and the device for the syslogs transaction log.

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 switch 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.

Installing auditing with *installsecurity*

The *$SYBASE/scripts* directory contains *installsecurity*, a script 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 Transact-SQL disk init and create database commands. For example:

```
disk init name = "auditdev",
    physname = "/dev/dsk/c2d0s4",
    size = "10"
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:

```
cd $SYBASE/scripts
setenv DSQUERY server_name
isql -Usa -Ppassword -Sserver_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*.

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

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 sybsecurity database.
3. Install sybsecurity again using the installation procedure described in either:
   - The configuration documentation for your platform.
   - "Installing auditing with installsecurity" on page 397.
4. During the installation process, be sure to place the sybsecurity database on one or more devices, separate from the master device.

Moving *sybsecurity* and saving global audit settings

v **To move the *sybsecurity* database and save the global audit settings**

1. Dump the sybsecurity database:

   ```
   dump database sybsecurity to "/remote/sec_file"
   ```

2. Drop the sybsecurity database:

   ```
   drop database sybsecurity
   ```

3 Initialize the first device on which you want to place the sybsecurity database:

```
disk init name = "auditdev",
    physname = "/dev/dsk/c2d0s4",
    size = "10M"
```

4 Initialize the device where you want to place the security log:

```
disk init name = "auditlogdev",
    physname = "/dev/dsk/c2d0s5",
    size = "2M"
```

5 Create the new sybsecurity database:

```
create database sybsecurity on auditdev
    log on auditlogdev
```

6 Load the contents of the old sybsecurity database into the new database. The global audit settings are preserved:

```
load database sybsecurity from "/remote/sec_file"
```

7 Run online database, which will upgrade sysaudits and sysauditoptions if necessary:

```
online database sybsecurity
```

8 Load the auditing system procedures using the configuration documentation for your platform.

v **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 408.

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 sp_configure.
- Archive the audit table that is almost full using the insert and select commands.

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 sp_configure, using the following syntax, where *n* is an integer that determines the new current audit table:

```
sp_configure "current audit table", n
  [, "with truncate"]
```

The valid values for *n* are:

- 1 means sysaudits_01, 2 means sysaudits_02, and so forth.
- 0 tells Adaptive Server to automatically set the current audit table to the next table. For example, if your installation has three audit tables, sysaudits_01, sysaudits_02, and sysaudits_03, Adaptive Server sets the current audit table to:
    - 2 if the current audit table is sysaudits_01
    - 3 if the current audit table is sysaudits_02

- 1 if the current audit table is sysaudits_03

The with truncate option specifies that 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, sp_configure fails.

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 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.

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:

   ```
   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:

```
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:

```
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.

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
```

In addition, you can get 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
go
```

The sample threshold procedure audit_thresh receives control when fewer than 250 free pages remain in the current audit table.

For more information about adding threshold procedures, see Chapter 31, "Managing Free Space with Thresholds."

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 memory requirement for a single audit record is 424 bytes. The default size for the audit queue is 100 records, which requires approximately 42K.

To set the size of the audit queue, use sp_configure. The syntax is:

```
sp_configure "audit queue size", [value]
```

value is the number of records that the audit queue can hold. The minimum value is 1, and the maximum is 65,535. For example, to set the audit queue size to 300, execute:

```
sp_configure "audit queue size", 300
```

For more information about setting the audit queue size and other configuration parameters, see Chapter 4, "Setting Configuration Parameters."

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. You can 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.

To set this configuration parameter, use sp_configure. 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 will be 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 on, 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 400, 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-chance 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 would be 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 402 for more information.

Adaptive Server does not supply a default procedure, but Chapter 31, “Managing Free Space with Thresholds” contains examples of last-chance threshold procedures. The procedure should execute the dump transaction command, which truncates the log. When the transaction log reaches the last-chance threshold point, any transaction that is running is suspended until space is available. The suspension occurs because the option abort xact when log is full is always set to FALSE for the sybsecurity database. You cannot change this option.

With the trunc log on chkpt option off, you can use standard backup and recovery procedures for the sybsecurity database, but be aware that the audit tables in the restored database may not be in sync with their status at the time of a device failure.

Enabling and disabling auditing

To enable or disable auditing, use sp_configure with the auditing configuration parameter. 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 11-6 on page 424.

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 will be 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.

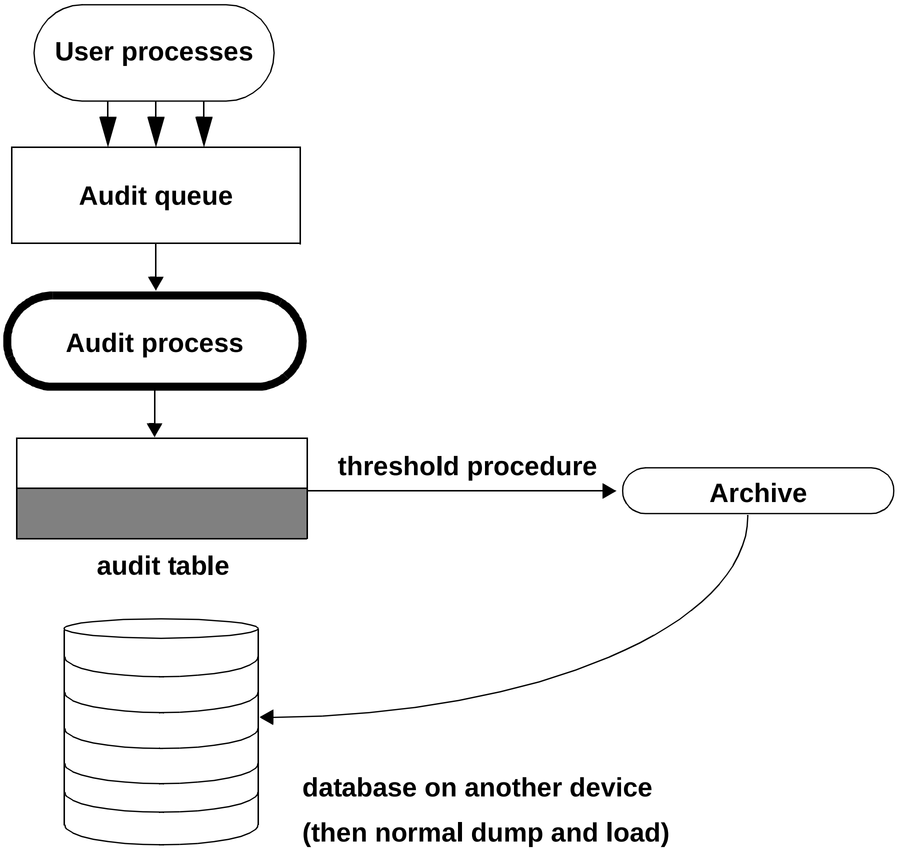
For *non-production* systems, where the loss of a small number of audit records may be acceptable, you can use a single table for auditing, if you cannot spare the additional disk space for multiple audit tables, or you do not have additional devices to use.

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 11-2 shows how the auditing process works with a single audit table.

Figure 11-2: Auditing with a single audit table



Establishing and managing single-table auditing

Table 11-2 provides an overview of managing single-table auditing.

Table 11-2: Auditing process for single-table auditing

Action	Description	See
1. Install auditing.	Installation of auditing, which involves setting the number of audit tables and assigning devices for the audit trail and the syslogs transaction log in the sybsecurity database.	The the installation documentation for your platform.

Action	Description	See
2. Set up the audit process to manage the audit trail.	Writing and establishing a threshold procedure that receives control when the audit table is nearly full. The procedure automatically writes the contents of the audit table to another table, and then truncates the audit table. In addition, this step involves setting the audit queue size and suspend audit when device full configuration parameters.	• "Establishing and managing single-table auditing" on page 409. • "Threshold procedure for single-table auditing" on page 410.
3. Set up the audit process to manage the syslogs transaction log in the sybsecurity database.	Determining how to handle the syslogs transaction log in the sybsecurity database. The task includes determining the setting of the trunc log on chkpt database option and establishing a last-chance threshold procedure for syslogs if trunc log on chkpt is off.	"Setting up transaction log management" on page 406.
4. Set auditing options.	Using sp_audit to establish the events to be audited. **Note** No audit records are generated until auditing is turned on with sp_configure.	"Setting global auditing options" on page 412
5. Enable auditing.	Using sp_configure to turn on the auditing configuration parameter. Adaptive Server begins writing audit records for audited events to the current audit table.	"Enabling and disabling auditing" on page 407

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:

```
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 402.

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, it is possible that the audit table can 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 405.

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 goes 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 will be unable to execute commands.

Recovering when the current audit table is full

If the current audit device and the audit queue becomes 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 401. 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.

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 the *Reference Manual*.

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 407.

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 sybsecurity..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.

Table 11-3 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 of all options is off.

Table 11-3: Auditing options, requirements, and examples

Option (option type)	***login_name***	***object_name***	**Database to be in to set the option**	**Command or access being audited**
adhoc	all	all	Any	Allows users to use sp_addauditrecord
(user-specific)	Example: `sp_audit "adhoc", "all", "all", "on"` (Enables ad hoc user-defined auditing records.)			

Option (option type)	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
all (user-specific)	A login name or role	all	Any	All actions of a particular user or by users with a particular role active
	Example `sp_audit "all", "sa_role", "all", "on"` (Turns auditing on for all actions in which the sa_role is active.)			
alter (database-specific)	all	Database to be audited	Any	alter database, alter table
	Example `sp_audit @option = "alter", @login_name = "all", @object_name = "master", @setting = "on"` (Turns auditing on for all executions of alter database and alter table in the master database.)			
bcp (database-specific)	all	Database to be audited	Any	bcp in
	Example `sp_audit "bcp", "all", "pubs2"` (Returns the status of bcp auditing in the pubs2 database. 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
	Example `sp_audit "bind", "all", "planning", "off"` (Turns bind auditing off for the planning database.)			
cmdtext (user-specific)	A login name, role, or "all" for all users in the database	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.)
	Example `sp_audit "cmdtext", "sa", "all", "off"` (Turns text auditing off for Database Owners.)			
create (database-specific)	all	Database to be audited	Any	create database, create table, create procedure, create trigger, create rule, create default, sp_addmessage, create view, create index, create function
	Note Specify master for *object_name* if you want to audit create database. You will also be auditing the creation of other objects in master. **Example** `sp_audit "create", "all", "planning", "pass"` (Turns on auditing of successful object creations in the planning database. The current status of auditing create database is not affected because you did not specify the master database.)			

Option (option type)	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
dbaccess (database-specific)	all	Database to be audited	Any	Any access to the database from another database
	Example `sp_audit "dbaccess", "all", "project", "on"` (Audits all external accesses to the project database.)			
dbcc (global)	all	all	Any	All dbcc commands that require permissions
	Example `sp_audit "dbcc", "all", "all", "on"` (Audits all executions of the dbcc command.)			
delete (object-specific)	all	Name of the table or view to be audited, or default view or default table	The database of the table or view (except tempdb)	delete from a table, delete from a view
	Example `sp_audit "delete", "all", "default table", "on"` (Audits all delete actions for all future tables in the current database.)			
disk (global)	all	all	Any	disk init, disk refit, disk reinit, disk mirror, disk unmirror, disk remirror, disk resize
	Example `sp_audit "disk", "all", "all", "on"` (Audits all disk actions for the server.)			
drop (database-specific)	all	Database to be audited	Any	drop database, drop table, drop procedure, drop index, drop trigger, drop rule, drop default, sp_dropmessage, drop view, drop function
	Example `sp_audit "drop", "all", "financial", "fail"` (Audits all drop commands in the financial database that fail permission checks.)			
dump (database-specific)	all	Database to be audited	Any	dump database, dump transaction
	Example `sp_audit "dump", "all", "pubs2", "on"` (Audits dump commands in the pubs2 database.)			
errors (global)	all	all	Any	Fatal error, non-fatal error
	Example `sp_audit "errors", "all", "all", "on"` (Audits errors throughout the server.)			

Option (option type)	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
exec_procedure (object-specific)	all	Name of the procedure to be audited or default procedure	The database of the procedure (except tempdb)	execute
	Example `sp_audit "exec_procedure", "all", "default procedure", "off"` (Turns automatic auditing off for new procedures in the current database.)			
exec_trigger (object-specific)	all	Name of the trigger to be audited or default trigger	The database of the trigger (except tempdb)	Any command that fires the trigger
	Example `sp_audit "exec_trigger", "all", "trig_fix_plan",  "fail"` (Audits all failed executions of the trig_fix_plan trigger in the current database.)			
func_dbaccess (database-specific)	all	Name of the database you are auditing	Any	Access to the database using the following functions: curunreserved_pgs, db_name, db_id, lct_admin, setdbrepstat, setrepstatus, setrepdefmode, is_repagent_enabled, rep_agent_config, rep_agent_admin
	Example `sp_audit @option="func_dbaccess", @login_name="all", @object_name = "strategy", @setting = "on"` (Audits accesses to the strategy database via built-in functions.)			
func_obj_access (object-specific)	all	Name of any object that has an entry in sysobjects	Any	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
	Example `sp_audit @option="func_obj_access", @login_name="all", @object_name = "customer", @setting = "on"` (Audits accesses to the customer table via built-in functions.)			
grant (database-specific)	all	Name of the database to be audited	Any	grant
	Example `sp_audit @option="grant", @login_name="all", @object_name = "planning", @setting = "on"` (Audits all grants in the planning database.)			

Option (option type)	***login_name***	***object_name***	**Database to be in to set the option**	**Command or access being audited**
insert (object-specific)	all	Name of the view or table to which you are inserting rows, or default view or default table	The database of the object (except tempdb)	insert into a table, insert into a view
	Example `sp_audit "insert", "all", "dpt_101_view", "on"` (Audits all inserts into the dpt_101_view view in the current database.)			
install	all	Database to be audited	Any	install java
	Example `sp_audit "install", "all", "planning", "on"` (Audits the installation of java classes in database planning)			
load (database-specific)	all	Database to be audited	Any	load database, load transaction
	Example `sp_audit "load", "all", "projects_db", "fail"` (Audits all failed executions of database and transaction loads in the projects_db database.)			
login (global)	all	all	Any	Any login to Adaptive Server
	Example `sp_audit "login", "all", "all", "fail"` (Audits all failed attempts to log in to the server.)			
logout	all	all	Any	Any logout from Adaptive Server
	Example `sp_audit "logout", "all", "all", "off"` (Turns auditing off of logouts from the server.)			
mount (global)	all	all	Any	mount database
	Example `sp_audit "mount", "all", "all", "on"` (Audits all mount database commands issued.)			
quiesce (global)	all	all	Any	quiesce database
	Example `sp_audit "quiesce", "all", "all", "on"` (Turns auditing on for quiesce database commands.)			
reference (object-specific)	all	Name of the view or table to which you are inserting rows, or default view or default table	Any	create table, alter table
	Example `sp_audit "reference", "all", "titles", "off"` (Turns off auditing of the creation of references to the titles table.)			

Option (option type)	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
remove (global)	all	all	Any	Audits the removal of Java classes
	Example `sp_audit "remove", "all", "all", "on"` (Audits the removal of Java classes in all databases.)			
revoke (database-specific)	all	Database to be audited	Any	revoke
	Example `sp_audit "revoke", "all", "payments_db", "off"` (Turns off auditing of the execution of revoke in the payments_db database.)			
rpc (global)	all	all	Any	Remote procedure calls (either in or out)
	Example `sp_audit "rpc", "all", "all", "on"` (Audits all remote procedure calls out of or into the server.)			
security (global)	all	all	Any	Server-wide security-relevant events. See the "security" option in Table 11-6.
	Example `sp_audit "security", "all", "all", "on"` (Audits server-wide security-relevant events in the server.)			
select (object-specific)	all	Name of the view or table to which you are inserting rows, or default view or default table	The database of the object (except tempdb)	select from a table, select from a view
	Example `sp_audit "select", "all", "customer", "fail"` (Audits all failed selects from the customer table in the current database.)			
setuser (database-specific)	all	all	Any	setuser
	Example `sp_audit "setuser", "all", "projdb", "on"` (Audits all executions of setuser in the projdb database.)			
table_access (user-specific)	Name of the login to be audited, or all if all users are to be audited.	all	Any	select, delete, update, or insert access in a table
	Example `sp_audit "table_access", "smithson", "all", "on"` (Audits all table accesses by the login named "smithson".)			
truncate (database-specific)	all	Database to be audited	Any	truncate table
	Example `sp_audit "truncate", "all", "customer", "on"` (Audits all table truncations in the customer database.)			

Option (option type)	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
unbind (database-specific)	all	Database to be audited	Any	sp_unbindefault, sp_unbindrule, sp_unbindmsg
	Example `sp_audit "unbind", "all", "master", "fail"` (Audits all failed attempts of unbinding in the master database.)			
unmount	all	all	Any	unmount database
	Example `sp_audit "unmount", "all", "projects", "on"` (Audits all attempts by users to mount the projects table in the current database.)			
update (object-specific)	all	Name specifying the object to be audited, default table or default view	The database of the object (except tempdb)	update to a table, update to a view
	Example `sp_audit "update", "all", "projects", "on"` (Audits all attempts by users to update the projects table in the current database.)			
view_access (user-specific)	Login name of the user to be audited, or all to audit all users	all	Any	select, delete, insert, or update to a view
	Example `sp_audit "view_access", "joe", "all", "off"` (Turns off view auditing of user “joe”.)			

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. To set object-specific auditing options, you must be in the object’s database before you execute sp_audit:

```
sp_audit "security", "all", "all", "fail"
```

For this example, execute:

```
use company_operations
go
sp_audit "delete", "all", "projects", "fail"
go
sp_audit "delete", "all", "default table",
"fail"
```

```
go
```

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 the *Reference Manual*.

Adding user-specified records to the audit trail

sp_addauditrecord allows users to enter comments into the audit trail. The syntax is:

```
sp_addauditrecord [text] [, db_name] [, obj name]
    [, owner_name] [, dbid] [, objid]
```

All the parameters are optional:

- *text* – is the text of the message that you want to add to the extrainfo audit table.
- *db_name* – is the name of the database referred to in the record, which is inserted into the dbname column of the current audit table.
- *obj_name* – is the name of the object referred to in the record, which is inserted into the objname column of the current audit table.
- *owner_name* – is the owner of the object referred to in the record, which is inserted into the objowner column of the current audit table.
- *dbid* – is an integer value representing the database ID number of db_name, which is inserted into the dbid column of the current audit table. Do not place it in quotes.
- *objid* – is an integer value representing the object ID number of obj_name. Do not place it in quotes. *objid* is inserted into the objid column of the current audit table.

You can use sp_addauditrecord if:

- You have execute permission on sp_addauditrecord.
- The auditing configuration parameter was activated with sp_configure.

- The adhoc auditing option was enabled with sp_audit.

By default, only a System Security Officer and the Database Owner of sybsecurity can use sp_addauditrecord. Permission to execute it may be granted to other users.

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:

```
sp_addauditrecord "I gave A. Smith permission to view
the payroll table in the corporate database. This
permission was in effect from 3:10 to 3:30 pm on
9/22/92.", "corporate", "payroll", "dbo", 10,
1004738270
```

The following example inserts information only into the extrainfo and dbname columns of the current audit table:

```
sp_addauditrecord @text="I am disabling auditing
briefly while we reconfigure the system",
@db_name="corporate"
```

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 400, the audit data is automatically archived to one or more tables in another database. For example, assume that the audit data resides in a table called audit_data in the audit_db database. To select audit records for tasks performed by “bob” on July 5, 1993, execute:

```
use audit_db
go
select * from audit_data
   where loginname = "bob"
   and eventtime like "Jul 5% 93"
go
```

This command requests audit records for commands performed in the pubs2 database by users with the System Security Officer role active:

```
select * from audit_data
   where extrainfo like "%sso_role%"
   and dbname = "pubs2"
go
```

This command requests audit records for all table truncations (event 64):

```
select * from audit_data
   where event = 64
go
```

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 11-4 describes the columns in all audit tables.

Table 11-4: Columns in each audit table

Column name	Datatype	Description
event	smallint	Type of event being audited. See Table 11-6 on page 424.
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.

Column name	Datatype	Description
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.
extrainfo	varchar(255) null	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 423.

Reading the *extrainfo* column

The extrainfo column contains a sequence of data separated by semicolons. The data is organized in the following categories.

Table 11-5: Information in the extrainfo column

Position	Category	Description
1	Roles	A list of active roles, separated by blanks.
2	Keywords or Options	The name of the keyword or option that was used for the event. For example, for the alter table command, the add column or drop constraint options might have been used. If multiple keywords or options are listed, they are separated by commas.
3	Previous value	If the event resulted in the update of a value, this item contains the value prior to the update.
4	Current value	If the event resulted in the update of a value, this item contains the new value.
5	Other information	Additional security-relevant information that is recorded for the event.
6	Proxy information	The original login name if the event occurred while a set proxy was in effect.
7	Principal name	The principal name from the underlying security mechanism if the user's login is the secure default login, and the user logged into Adaptive Server via unified login. The value of this item is NULL if the secure default login is not being used.

This example shows an extrainfo column entry for the event of changing an auditing configuration parameter.

```
sso_role; SETCONFIG; 1; 0; 261; ralph; ;
```

This entry indicates that a System Security Officer changed the value of a configuration parameter from 1 to 0. The fifth catagory (Other Information) indicates that configuration parameter number 261 (which is the suspend audit when device full configuration parameter) was changed. The sixth category (Proxy Information) 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 11-6 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 11-5.

Table 11-6: Values in event and extrainfo columns

Audit Option	Command or access to be audited	event	Information in extrainfo
(Automatically audited event not controlled by an option)	Enabling auditing with: sp_configure auditing	73	—
(Automatically audited event not controlled by an option)	Disabling auditing with: sp_configure auditing	74	—
adhoc	User-defined audit record	1	extrainfo is filled by the text parameter of sp_addauditrecord
alter	alter database	2	*Keywords or options:* alter maxhold alter size
	alter table	3	*Keywords or options:* add column drop column replace column add constraint drop constraint
bcp	bcp in	4	—
bind	sp_bindefault	6	*Other information:* Name of the default
	sp_bindmsg	7	*Other information:* Message ID
	sp_bindrule	8	*Other information:* Name of the rule
cmdtext	All commands	92	Full text of command, as sent by the client

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	—
	create trigger	12	—
	create view	16	—
	create index	104	*Other information*: Name of the index
	create function	97	—
	sp_addmessage	17	*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
disk	disk init	20	*Keywords or options:* disk init *Other information:* Name of the disk
	disk mirror	23	*Keywords or options:* disk mirror *Other information:* Name of the disk
	disk refit	21	*Keywords or options:* disk refit *Other information:* Name of the disk
	disk reinit	22	*Keywords or options:* disk reinit *Other information:* Name of the disk
	disk remirror	25	*Keywords or options:* disk remirror *Other information:* Name of the disk
	disk unmirror	24	*Keywords or options:* disk unmirror *Other information:* Name of the disk
	disk resize	100	*Keywords or options:* disk resize *Other information:* Name of the disk

Audit Option	Command or access to be audited	event	Information in extrainfo
drop	drop database	26	—
	drop default	31	—
	drop procedure	28	—
	drop table	27	—
	drop trigger	29	—
	drop rule	30	—
	drop view	33	—
	drop index	105	*Other information*: Index name
	drop function	98	—
	sp_dropmessage	32	*Other information:* Message number
dump	dump database	34	—
	dump transaction	35	—
errors	Fatal error	36	*Other information:* *Error number*.*Severity*.*State*
	Non-fatal error	37	*Other information:* *Error number*.*Severity*.*State*
exec_procedure	Execution of a procedure	38	*Other information:* All input parameters
exec_trigger	Execution of a trigger	39	—
func_obj_access, func_dbaccess	Accesses to objects and databases via Transact-SQL functions	86	—
grant	grant	40	—
insert	insert into a table	41	*Keywords or options:* • If insert is used: insert • If select into is used: insert into followed by the fully qualified object name
	insert into a view	42	*Keywords or options*: insert
install	install	93	—
load	load database	43	—
	load transaction	44	—
login	Any login to the server	45	*Other information:* Host name and IP adress of the machine from which the login was done.
logout	Any logouts from the server	46	*Other information:* Host name
mount	mount database	101	—
quiesce	quiesce database	96	—
reference	Creation of references to tables	91	*Keywords or options*: reference *Other information:* Name of the referencing table

Audit Option	Command or access to be audited	event	Information in extrainfo
remove	remove java	94	—
revoke	revoke	47	—
roles	create role, drop role, alter role, grant role, revoke role	85	—
rpc	Remote procedure call from another server	48	*Keywords or options:* Name of client program *Other information:* Server name, host name of the machine from which the RPC was done.
	Remote procedure call to another server	49	*Keywords or options:* Procedure name
security	connect to (CIS only)	90	*Keywords or options:* connect to
	kill (CIS only)	89	*Keywords or options:* kill
	online database	83	—
	proc_role function (executed from within a system procedure)	80	*Other information:* Required roles
	Regeneration of a password by an SSO	76	*Keywords or options:* Setting SSO password *Other information:* Login name
	Role toggling	55	*Previous value:* on or off *Current value:* on or off *Other information:* Name of the role being set
	Server start	50	*Other information:* -d*masterdevicename* -i*interfaces file path* -S*servername* -e*errorfilename*
	Server shutdown	51	*Keywords or options:* shutdown
	set proxy or set session authorization	88	*Previous value:* Previous suid *Current value:* New suid
	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
	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

Audit Option	Command or access to be audited	event	Information in extrainfo
	built-in functions	86	*Keywords or options:* Name of function
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_access	delete	18	*Keywords or options:* delete
	insert	41	*Keywords or options:* insert
	select	62	*Keywords or options:* select into select readtext
	update	70	*Keywords or options:* update writetext
truncate	truncate table	64	—
unbind	sp_unbindefault	67	—
	sp_unbindmsg	69	—
	sp_unbindrule	68	—
unmount	unmount database	102	—
update	update to a table	70	*Keywords or options:* update writetext
	update to a view	71	*Keywords or options:* update writetext

Audit Option	Command or access to be audited	event	Information in extrainfo
view_access	delete	19	*Keywords or options:* delete
	insert	42	*Keywords or options:* insert
	select	63	*Keywords or options:* select into select readtext
	update	71	*Keywords or options:* update writetext

CHAPTER 12 Managing User Permissions

This chapter describes the use and implementation of user permissions.

Overview

Discretionary access controls (DAC) allow you to restrict access to objects and commands based on a user's identity or group membership. 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.

System Administrators operate outside the DAC system and have access permissions on all database objects at all times. System Security Officers can always access the audit trail tables in the sybsecurity 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 481.

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 these commands to give users permission to create databases, to create objects within a database, execute certain commands such as set proxy, and to access specified tables, views, and columns. 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, 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 491.

Types of users and their privileges

Adaptive Server's discretionary access control system recognizes the following types of users:

- System Administrators
- System Security Officers
- Operators
- Database Owners
- Database object owners
- Other users (also known as "public")

System Administrator privileges

System Administrators:

- Handle tasks that are not specific to applications
- Work outside Adaptive Server's 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 has permission to 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.

The fact that a System Administrator operates 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, and the disk mirroring 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.

In addition, System Administrators are responsible for dropping logins and can lock and unlock logins. System Security Officers share login management responsibilities with System Administrators. System Security Officers are responsible for adding logins and can also lock and unlock logins.

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.

System Security Officer privileges

System Security Officers perform security-sensitive tasks in Adaptive Server, including:

- Granting the System Security Officer and Operator roles
- Administering the audit system
- Changing passwords
- Adding new logins
- Locking and unlocking login accounts
- Creating and granting user-defined roles
- Administering network-based security
- Granting permission to use the set proxy or set session authorization commands

The System Security Officer can *access* any database—to enable auditing —but, in general, has no special permissions on database objects. 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 damage inadvertently done to the protection system by a user. For example, if the Database Owner forgets his or her password, a System Security Officer can change the password to allow the Database Owner to log in.

System Security Officers can also create and grant user-defined roles to users, other roles, or groups. For information about creating and granting user-defined roles, see "Creating and assigning roles to users" on page 356.

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

Database Owner privileges

Database Owners and System Administrators are the only users who can grant object creation permissions to other users. 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 these commands is automatically granted to the Database Owner and cannot be transferred to other users:

- checkpoint
- dbcc
- drop database
- dump database
- dump transaction
- grant (object creation permissions)
- load database
- load transaction
- revoke (object creation permissions)
- setuser

Database Owners can grant permission to use these commands to other users:

- create default
- create procedure

- create rule
- create table
- create view
- grant (permissions on system tables)
- grant (select, insert, delete, update, references, and execute permissions on database objects)
- revoke (permissions on system tables)
- revoke (select, insert, delete, update, references, and execute permissions on database objects)

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. By default, when Adaptive Server is installed, the *installmodel* script grants select access to "public" (all users) for most system tables and for most fields in the tables. However, no access is given for some system tables, such as systhresholds, and no access is given for certain fields in other system tables. For example, all users, by default, can select all columns of sysobjects except audflags.

To determine the current permissions for a particular system table, execute:

sp_helprotect *system_table_name*

For example, to check the permissions of systhresholds in *your_database*, execute:

```
use your_database
go
sp_helprotect systhresholds
go
```

The default situation is that no users—including Database Owners—can modify the system tables directly. Instead, the system procedures supplied with Adaptive Server modify the system tables. This helps guarantee integrity.

Warning! Although does provide a mechanism that allows you to modify system tables, Sybase strongly recommends that you do not do so.

Permissions on system procedures

Permissions on system procedures are set 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.

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.

It is a good idea to 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 will 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
```

To transfer aliases and their permissions to the new "dbo," add the value true parameters.

Note You cannot change the ownership of the master database and should not change the ownership of any other system databases.

Database object owner privileges

A user who creates a database object (a table, view, 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 creation permissions default to the owner of a table and cannot be transferred to other users:

- alter table
- drop table
- create index
- truncate table
- update statistics

Permission to use the grant and revoke commands to grant specific users select, insert, update, delete, references, 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, or default—defaults to the object owner and cannot be transferred.

Privileges of other database users

At the bottom of the hierarchy are other database users. Permissions are granted to or revoked from them by object owners, Database Owners, users who were granted permissions, or a System Administrator. These users are specified by user name, group name, or the keyword public.

Granting and revoking permissions on database objects

Two types of permissions exist for objects:

- **Object access permissions** – for using the commands that access database objects. For more information, see "Granting and revoking object access permissions" on page 439.
- **Object creation permissions** –for creating objects. They can be granted only by a System Administrator or a Database Owner. For more information, see "Granting and revoking object creation permissions" on page 445.

Both types of permissions are controlled with the grant and revoke commands.

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.

Granting and revoking 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), who can grant them to other users.

Table 12-1 lists the types of object access permissions and the objects to which they apply.

Table 12-1: Permissions and the objects to which they apply

Permission	Object
select	Table, view, column
update	Table, view, column
insert	Table, view
delete	Table, view
references	Table, column
execute	Stored procedure

The references permission refers to referential integrity constraints that you can specify in an alter table or create table command. The other permissions refer to SQL commands. Object access permissions default to System Administrators and the object's owner, and can be granted to other users.

Use the grant command to grant object access permissions. The syntax is:

```
grant {all [privileges]| permission_list}
    on { table_name [(column_list)]
       | view_name[(column_list)]
       | stored_procedure_name}
    to {public | name_list | role_name}
    [with grant option]
```

Use the revoke command to revoke object access permissions. The syntax is:

```
revoke [grant option for]
    {all [privileges] | permission_list}
    on { table_name [(column_list)]
       | view_name [(column_list)]
       | stored_procedure_name}
    from {public | name_list | role_name}
    [cascade]
```

Notes on the keywords and parameters are as follows:

- all or all privileges specifies all permissions applicable to the specified object. All object owners can use all with an object name to grant or revoke permissions on their own objects. If you are granting or revoking permissions on a stored procedure, all is the same as execute.

Note insert and delete permissions do not apply to columns, so you cannot include them in a permission list (or use the keyword all) if you specify a column list.

- *permission_list* is the list of permissions that you are granting. If you name more than one permission, separate them with commas. Table 12-2 illustrates the access permissions that can be granted on each type of object:

Table 12-2: Object access permissions

Object	permission_list can include
Table or view	select, insert, delete, update, references. references applies to tables but not views; the other permissions apply to both tables and views.
Column	select, update, references
Stored procedure	execute

You can specify columns in the *permission_list* or the *column_list*, but not both.

- on specifies the object for which the permission is being granted or revoked. You can grant or revoke permissions for only one table, view, or stored procedure object at a time. You can grant or revoke permissions for more than one column at a time, but all the columns must be in the same table or view. You can only grant or revoke permissions on objects in your current database.
- public refers to the group “public,” which includes all Adaptive Server users. public means slightly different things for grant and revoke:
  - For grant, public includes the object owner. Therefore, if you have revoked permissions from yourself on your object, and later you grant permissions to public, you regain the permissions along with the rest of “public.”
  - For revoke, public excludes the owner.
- *name_list* includes:
  - Group names
  - User names

    - A combination of user and group names, each separated from the next by a comma

- *role_name* is an Adaptive Server system-defined or user-defined role. You can create and define a hierarchy of user-defined roles and grant them privileges based on the specific role granted. System-defined roles include sa_role (System Administrator), sso_role (System Security Officer), and oper_role (Operator). You cannot create or modify system-defined roles.
- with grant option in a grant statement allows the user(s) specified in *name_list* to grant the specified object access permission(s) to other users. If a user has with grant option permission on an object, that permission is not revoked when permissions on the object are revoked from public or a group of which the user is a member.
- grant option for revokes with grant option permissions, so that the user(s) specified in *name_list* can no longer grant the specified permissions to other users. If those other users have granted permissions to other users, you must use the cascade option to revoke permissions from them as well. The user specified in *name_list* retains permission to access the object, but can no longer grant access to other users. grant option for applies only to object access permissions, not to object creation permissions.
- The cascade option in a revoke statement removes the specified object access permissions from the user(s) specified in *name_list*, and also from any users they granted those permissions to.

You may only grant and revoke permissions on objects in the current database.

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 or until a System Administrator revokes the access. That is, if a System Administrator revokes access, the user is denied access even though other users have granted access.

Only a System Security Officer can grant or revoke permissions to create triggers. The Database Owner can create triggers on any user table. Users can only create triggers on tables that they own.

Permission to issue the create trigger command is granted to users by default.

When the System Security Officer revokes permission for a user to create triggers, a revoke row is added in the sysprotects table for that user. To grant permission to that user to issue create trigger, issue two grant commands: the first command removes the revoke row from sysprotects; the second inserts a grant row. If permission to create triggers is revoked, the user cannot create triggers even on tables that the user owns. Revoking permission to create triggers from a user affects only the database where the revoke command was issued.

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:

```
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.

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, recreate them with a different owner.

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 12-3 summarizes the required permissions.

Table 12-3: ANSI permissions for update and delete

	Permissions required: *set ansi_permissions off*	Permissions required: *set ansi_permissions on*
update	update permission on columns where values are being set	update permission on columns where values are being set and select permission on all columns appearing in the where clause select permission on all columns on the right side of the set clause
delete	delete permission on the table	delete permission on the table from which rows are being deleted and select permission on all columns appearing in the where clause

If ansi_permissions is on and you attempt to update or delete without having all the additional select permissions, the transaction is rolled back and you receive an error message. If this occurs, the object owner must grant you select permission on all relevant columns.

Examples of granting object access permissions

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

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

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

```
grant execute
on makelist
to harold
```

This statement grants permission to execute the custom stored procedure sa_only_proc to users who have been granted the System Administrator role:

```
grant execute
on sa_only_proc
to sa_role
```

This statement gives Aubrey permission to select, update, and delete from the authors table and to grant the same permissions to other users:

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

Examples of revoking object access permissions

These two statements both revoke permission for all users except the table owner to update the price and total_sales columns of the titles table:

```
revoke update
on titles (price, total_sales)
from public
revoke update(price, total_sales)
on titles
from public
```

This statement revokes permission from Clare to update the authors table and simultaneously revokes that permission from all users to whom she had granted that permission:

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

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

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

Granting and revoking object creation permissions

Object creation permissions regulate the use of commands that create objects. These permissions can be granted only by a System Administrator or a Database Owner.

The object creation commands are:

- create database
- create default
- create procedure

- create rule
- create table
- create view

The syntax for object creation permissions differs slightly from the syntax for object access permissions. The syntax for grant is:

```
grant {all [privileges] | command_list}
    to {public | name_list | role_name}
```

The syntax for revoke is:

```
revoke {all [privileges] | command_list}
    from {public | name_list | role_name}
```

where:

- all or all privileges – can be used only by a System Administrator or the Database Owner. When used by a System Administrator in the master database, grant all assigns all create permissions, including create database. If the System Administrator executes grant all from another database, all create permissions are granted except create database. When the Database Owner uses grant all, Adaptive Server grants all create permissions except create database, and prints an informational message.
- *command_list* – is the object creation permissions that you are granting or revoking. Separate commands with commas. The list can include create database, create default, create procedure, create rule, create table, and create view. create database permission can be granted only by a System Administrator, and only from within the master database.
- public – is all users except the Database Owner (who “owns” object creation permissions within the database).
- *name_list* – is a list of user or group names, separated by commas.
- *role_name* – is the name of an Adaptive Server system or user-defined role. You can create and define a hierarchy of user-defined roles and grant them privileges based on the specific role granted.

Examples of granting object creation permissions

The first example grants Mary and John permission to use create database and create table. Because create database permission is being granted, this command can only be executed by a System Administrator within the master database. Mary and John’s create table permission applies only to the master database.

```
grant create table, create database
to mary, john
```

This command grants permission to create tables and views in the current database to all users:

```
grant create table, create view
to public
```

Example of revoking object creation permissions

This example revokes permission to create tables and rules from “mary:”

```
revoke create table, create rule
from mary
```

Combining *grant* and *revoke* statements

You can assign specific permissions to specific users, or, if most users are going to be granted most privileges, it may be easier to assign all permissions to all users, and then revoke specific permissions from specific users.

For example, a Database Owner can grant all permissions on the titles table to all users by issuing:

```
grant all
on titles
to public
```

The Database Owner can then issue a series of revoke statements, for example:

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

grant and revoke statements are order-sensitive: in case of a conflict, the most recently issued statement supersedes all others.

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 will change, for all users, all previously issued permissions that conflict with the new regime.

The same grant and revoke statements issued in different orders can create entirely different situations. For example, the following set of statements leaves Jose, who belongs to the public group, without any select permission on titles:

```
grant select on titles(title_id, title) to jose
revoke select on titles from public
```

In contrast, the same statements issued in the opposite order result in only Jose having select permission and only on the title_id and title columns:

```
revoke select on titles from public
grant select on titles(title_id, title) to jose
```

When you use the keyword public with grant, you are including yourself. With revoke on object creation permissions, you are included in public unless you are the Database Owner. With revoke on object access permissions, you are included in public unless you are the object owner. You may want to deny yourself permission to use your own table, while giving yourself permission to access a view built on it. To do this, you must issue grant and revoke statements explicitly setting your permissions. You can reinstitute the permission with a grant statement.

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 12-4 lists the tasks related to roles, the role required to perform the task, and the command to use.

Table 12-4: Tasks, required roles, and commands to use

Task	Required role	Command
Grant the sa_role role	System Administrator	grant role
Grant the sso_role role	System Security Officer	grant role
Grant the oper_role role	System Security Officer	grant role
Grant user-defined roles	System Security Officer	grant role
Create role hierarchies	System Security Officer	grant role
Modify role hierarchies	System Security Officer	revoke role
Revoke system roles	System Security Officer	revoke role
Revoke user-defined roles	System Security Officer	revoke role

Granting roles

To grant roles to users or other roles, use:

```
grant role role_granted [{, role_granted}...]
    to grantee [{, grantee}...]
```

where:

- *role_granted* – is the role being granted. You can specify any number of roles to be granted.
- *grantee* – is the name of the user or role. You can specify any number of grantees.

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
```

Understanding *grant* and roles

You can 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:

- System Administrator
- System Security Officer
- Operator
- Any user-defined role

A role can only be granted to a login account or another role.

However, grant permission does not prevent users who do *not* have the specified role from being granted execute permission on a stored procedure. If you want to ensure, for example, that only System Administrators can successfully execute a stored procedure, use the proc_role system function within the stored procedure itself. See “Displaying information about roles” on page 378 for more information.

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 is still able to access sales when he has sso_role active because his role permissions override his individual permissions.

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.

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.

Revoking roles

Use revoke role to revoke roles from users and other roles:

```
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.

You cannot revoke a role from a user while the user is logged in.

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 DBO 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_ASM license option. Row-level access control is a dynamic option, so you do not need to reboot the SQL Server.

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:

```
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 458, and "Using the Application Context Facility" on page 463.

Syntax for access rules

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

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

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(20), 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:

```
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

AND access rules and OR access rules can be bound 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.

```
create access rule empid1_access
as @empid = 1

create access rule deptno1_access
as @deptid = 2
```

The following steps create OR access rules.

```
create or access rule name1_access
as @name = "smith"
```

```
create or access rule phone_access
as @phone = "9999"
```

Create table

This step creates a test table.

```
create table testtab1 (empno int, deptno int,name
char(10), phone char(4))
```

Bind rules to table

The following steps bind access rules to the test table columns.

```
sp_bindrule empid1_access, "testtab1.empno"
/*Rule bound to table column.*/
(return status = 0)
```

```
sp_bindrule deptno1_access,"testtab1.deptno"
/*Rule bound to table column.*/
```

```
(return status = 0)
```

```
sp_bindrule name1_access,"testtab1.name"
/*Rule bound to table column.*/
```

```
(return status = 0)
```

```
sp_bindrule phone_access,"testtab1.phone"
/*Rule bound to table column.*/

(return status = 0)
```

Insert data into table

The following steps insert values into the test table.

```
insert testtab1 values (1,1,"smith","3245")
(1 row affected)

insert testtab1 values(2,1,"jones","0283")
(1 row affected)

insert testtab1 values(1,2,"smith","8282")(1 row
affected)

insert testtab1 values(2,2,"smith","9999")

(1 row affected)
```

Access rule examples

The following examples show how access rules return specific rows containing information limited by access rules.

Example 1

This example returns information from two rows.

```
/* return rows when empno = 1 and deptno = 2
and ( name = "smith" or phone = "9999" )
*/

select * from testtab1

 empno       deptno      name       phone

------------ ----------- ---------- -----

           1           2 smith      8282

           1           2 jones      9999


(2 rows affected)

/* unbind access rule from specific column */

sp_unbindrule "testtab1.empno",NULL,"accessrule"

/*Rule unbound from table column.*/

(return status = 0)
```

Example 2

This example returns information from four rows.

```
/* return rows when deptno = 2 and ( name = "smith"
or phone = "9999" )*/

select * from testtab1

 empno       deptno      name       phone

 ----------- ----------- ---------- -----

           1           2 smith      8282

           2           2 smith      9999

           3           2 smith      8888

           1           2 jones      9999


(4 rows affected)


/* unbind all deptno rules from specific column */

sp_unbindrule "testtab1.deptno",NULL,"all"
/*Rule unbound from table column.*/

(return status = 0)
```

Example 3

This example returns information from six rows.

```
/* return the rows when name = "smith" or phone = "9999"
*/

select * from testtab1

 empno       deptno      name       phone

 ----------- ----------- ---------- -----

           1           1 smith      3245

           1           2 smith      8282

           2           2 smith      9999

           3           2 smith      8888

           1           2 jones      9999

           2           3 jones      9999
```

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 bulk copy utility (bcp). Adaptive Server cannot disable access rules, as it does with alter table, because bcp can be used by any user 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:

```
import java.sql.*;
import java.util.*;

public class sec_class {
static String _url = "jdbc:sybase:asejdbc";
public static int GetSecVal(int c1)
{
try
{
PreparedStatement pstmt;
ResultSet rs = null;
Connection con = null;
    int pno_val;

pstmt = null;
```

```
Class.forName("sybase.asejdbc.ASEDriver");
con = DriverManager.getConnection(_url);

if (con == null)
{
return (-1);
}

pstmt = con.prepareStatement("select classification
from sec_tab where id = ?");

if (pstmt == null)
{
return (-1);
}

pstmt.setInt(1, c1);

rs = pstmt.executeQuery();

rs.next();

pno_val = rs.getInt(1);

rs.close();

pstmt.close();

con.close();

return (pno_val);

}
catch (SQLException sqe)
{
return(sqe.getErrorCode());
}
catch (ClassNotFoundException e)
{

System.out.println("Unexpected exception : " +
e.toString());
System.out.println("\nThis error usually indicates that
" + "your Java CLASSPATH environment has not been set
properly.");
```

```
e.printStackTrace();
return (-1);
}
catch (Exception e)
{
System.out.println("Unexpected exception : " +
e.toString());
e.printStackTrace();
return (-1);
}
}
}
import java.sql.*;
import java.util.*;

public class sec_class {
static String _url = "jdbc:sybase:asejdbc";
public static int GetSecVal(int c1)
{
try
{
PreparedStatement pstmt;
ResultSet rs = null;
Connection con = null;
    int pno_val;

pstmt = null;

Class.forName("sybase.asejdbc.ASEDriver");
con = DriverManager.getConnection(_url);

if (con == null)
{
return (-1);
}

pstmt = con.prepareStatement("select classification
from sec_tab where id = ?");

if (pstmt == null)
{
return (-1);
}

pstmt.setInt(1, c1);
```

```
rs = pstmt.executeQuery();

rs.next();

pno_val = rs.getInt(1);

rs.close();

pstmt.close();

con.close();

return (pno_val);

}
catch (SQLException sqe)
{
return(sqe.getErrorCode());
}
catch (ClassNotFoundException e)
{

System.out.println("Unexpected exception : " +
e.toString());
System.out.println("\nThis error usually indicates that
" + "your Java CLASSPATH environment has not been set
properly.");
e.printStackTrace();
return (-1);
}
catch (Exception e)
{
System.out.println("Unexpected exception : " +
e.toString());
e.printStackTrace();
return (-1);
}
}
}
```

After compiling the Java code, you can run the same program from isql, as follows.

For example:

```
javac sec_class.java
jar cufo sec_class. jar sec_class.class
```

```
installjava -Usa -Password -
f/work/work/FGAC/sec_class.jar -
-D testdb
```

From isql:

```
/*to create new user datatype class_level*/
sp_addtype class_level, int
/*to create the sample secure data table*/
create table sec_data (c1 varchar(30),
c2 varchar(30),
c3 varchar(30),
clevel class_level)
/*to create the classification table for each user*/
create table sec_tab (userid int, clevel class-level
int)

insert into sec_tab values (1,10)
insert into sec_tab values (2,9)
insert into sec_tab values (3,7)
insert into sec_tab values (4,7)
insert into sec_tab values (5,4)
insert into sec_tab values (6,4)
insert into sec_tab values (7,4)

declare @v1 int
select @v1 = 5
while @v1 > 0
begin
insert into sec_data values('8', 'aaaaaaaaaa',
'aaaaaaaaaa', 8)
insert into sec_data values('7', 'aaaaaaaaaa',
'aaaaaaaaaa', 7)
insert into sec_data values('5', 'aaaaaaaaaa',
'aaaaaaaaaa', 5)
insert into sec_data values('5', 'aaaaaaaaaa',
'aaaaaaaaaa', 5)
insert into sec_data values('2', 'aaaaaaaaaa',
'aaaaaaaaaa', 2)
insert into sec_data values('3', 'aaaaaaaaaa',
'aaaaaaaaaa', 3)
select @v1 = @v1 -1
end
go

create access rule clevel_rule
@clevel <= sec_class.GetSecVal(suser_id())
```

```
go

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

sp_bindefault clevel_def, class_level
go

sp_bindrule clevel, class_level
go

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

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 four 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 12-5 on page 471. You can also create your own application contexts, as described in “Creating and using application contexts” on page 465.

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, 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 built-in functions is defined in a table created by the System Administrator, containing all logins for all tables. For more information about this table, see "Using login triggers" on page 473.

- set_appcontext() stores:

```
select set_appcontext ("titles", "rlac", "1")
```

- get_appcontext() supplies two parts of a context in a session, and retrieves the third:

```
select get_appcontext ("titles", "rlac")
------------------------
1
```

For more information on these functions and on list_appcontext and rm_appcontext, see "Creating and using application contexts" on page 465.

Granting and revoking

You can 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.

The use of built-in 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

Built-in 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 only affect users named in the table created by the System Administrator. For information about the table, see “Using login triggers” on page 473. 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
```

Creating and using application contexts

The following built-in functions are available for creating and maintaining application contexts. For more information than is given in this section, see *Reference Manual.*

- 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*")

- *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(2048).

Examples

This example creates an application context called CONTEXT1, with an attribute ATTR1 that has the value VALUE1:

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

This example shows an attempt to override the existing application context. The attempt fails, returning -1:

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

This example 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))
------------
0
```

This example shows the result when a user without appropriate permissions attempts to set the application context. The attempt fails, returning -1:

```
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 with 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.

get_appcontext ("*context_name*", "*attribute_name*")

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

This example shows VALUE1 returned for ATTR1:

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

ATTR1 does not exist in CONTEXT2:

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

This example shows the result when a user without appropriate permissions attempts to get the application context:

```
select get_appcontext("CONTEXT1", "ATTR2")
select permisssion 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*")

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

Examples

To use list_appcontext, the user must have appropriate permissions. For more information about permissions, see “Setting permissions for using application context functions” on page 464.

This example 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
```

This example 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.

rm_appcontext

Removes a specific application context, or all application contexts.

rm_appcontext ("*context_name*", "*attribute_name*")

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

The following three examples show how to remove an application context by specifying some or all attributes. Use an asterisk ("*") to remove all attributes in the specified context.

```
select rm_appcontext("CONTEXT1", "*")
---------
0
```

Use an asterisk ("*") to remove all the contexts and attributes.

```
select rm_appcontext("*", "*")
---------
0
```

This example shows a user attemptimg to remove a nonexistent context. The attempt fails, returning -1.

```
select rm_appcontext("NON_EXISTING_CTX", "ATTR2")
---------
-1
```

This example shows the result of a user without appropriate permissions attempting to remove an application context.

```
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 12-5: sys_session attributes and values

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

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 the user's security level. This solution uses access rules, with the Application Context Facility, to display only the rows that one of the users, Dave, sees.

There are five logins:

- Anne has security level 1.
- Bob has security level 1.
- Cassie has security level 2.
- Dave has security level 2.

- Ellie has security level 4.

Users should see only rows with a value in rlac that is less than or equal to their own security level. To accomplish this, create an access rule and apply ACF.

The rlac column is type integer, and appcontext arguments are type char.

```
create access rule rlac_rule as
    @value <= convert(int, get_appcontext("titles",
                "rlac"))

sp_bindrule rlac_rule, "titles.rlac"

/* log in as Dave and apply ACF value of 2*/

select set_appcontext("titles", "rlac", "2")

/*this value persists throughout the session*/
/*select all rows*/

select title_id, rlac from titles
---------------------
```

title_id	rlac
PC8888	1
BU1032	2
PS7777	1
PS3333	1
BU1111	2
PC1035	1
BU2075	2
PS2091	1
PS2106	1
BU7832	2
PS1372	1

```
(11 rows affected)
```

Using login triggers

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

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 creates a login trigger stored procedure in the Pubs2 database.

```
create loginproc as
   declare @appname varchar(20)
   declare @attr    varchar(20)
   declare @value      varchar(20)
   declare @retvalue   int
declare apctx cursor for
 select appname, attr, value from
 pubs2.dbo.lookup where login = suser_name()
open apctx
fetch apctx into @appname, @attr, @value

While (@@sqlstatus = 0)
   begin
      select f@retval =
         set_appcontext (rtrim (@appname),
         rtrim(@attr), rtrim(@value))
   fetch apctx into @appname, @attr, @value
 end
go

grant execute on loginproc to public
go
```

To associate a specific user with the login trigger, run sp_modifylogin 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 sp_modifylogin. The syntax is:

sp_modifylogin <*login_name*>, "login script", <*sproc_name* >

- *login_name* – the user's login name.
- "login script" – type in as shown; "login script" tells sp_modifylogin that the next parameter, "sproc_name", is a login trigger.
- *sproc_name* – the name of the stored procedure configured as a login trigger for this user.

Run this procedure 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.

Configuring the login trigger

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:

```
sp_modifylogin my_login, "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:

```
sp_modifylogin my_login, "login script", NULL
```

To change the login trigger to a different stored procedure, enter:

```
sp_modifylogin my_login, "login script", diff_proc
```

Displaying the login trigger

To display the current login trigger, use sp_displaylogin:

```
sp_displaylogin my_login
go
(....)
Default Database: my_db
Default Language:
Auto Login Script: my_proc
....
```

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 will 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.

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 were designed as a 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:

```
create procedure limit_user_sessions
as
declare @cnt int,
   @limit int,
   @loginname varchar(32)

select @limit = 2 -- max nr. of concurrent logins

/* determine current #sessions */
select @cnt = count(*)
from master.dbo.sysprocesses
where suid = suser_id()
```

```
/* check the limit */
if @cnt > @limit
begin

    select @loginname = suser_name()
    print "Aborting login [%1!]: exceeds session
        limit [%2!]",
        @loginname, @limit
     /* abort this session */
     select syb_quit()
end
go

grant exec on limit_user_sessions to public
go
```

2 As System Security Officer, configure this stored procedure as a login trigger for user "bob":

```
sp_modifylogin "bob", "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:
ct_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]
```

Enforcing timed-based restrictions

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 Administraor, create this table:

```
create table access_times (
suid int not null,
dayofweek tinyint,
shiftstart time,
shiftend time)
```

2 As System Administrator, insert the following rows in table access_times. These rows indicate that user “bob” is allowed to log into 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

```
insert into access_times
select suser_id(‘bob’), 1, ‘9:00’, ‘17:00’
go
insert into access_times
select suser_id(‘mark’), 2, ‘9:00’, ‘17:00’
go
```

3 As System Administrator, create the limit_access_time stored procedure, which references the access_time table to determine if login access should be granted:

```
create procedure limit_access_time as
declare @curdate date,
    @curdow tinyint,
    @curtime time,
    @cnt int,
    @loginname varchar(32)

-- setup variables for current day-of-week, time
select @curdate = current_date()
select @curdow = datepart(cdw,@curdate)
select @curtime = current_time()
select @cnt = 0

-- determine if current user is allowed access
select @cnt = count(*)
from access_times
where suid = suser_id()
and dayofweek = @curdow
and @curtime between shiftstart and shiftend

if @cnt = 0
begin
   select @loginname = suser_name()
   print "Aborting login [%1!]: login attempt past
     normal working hours", @loginname

   -- abort this session
   return -4
```

```
end
go

grant exec on limit_access_time to public
go
```

4 As System Security Officer, configure the limit_access_time stored procedure as a login trigger for users “bob” and “mark”:

```
sp_modifylogin "bob", "login script",
"limit_access_time"
go
sp_modifylogin "mark", "login script",
"limit_access_time"
go
```

5 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 Adpative 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
```

6 The following message is logged in the errorlog:

```
(...) 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,” you need 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 use a login trigger to set session-specific options, such as set nocount on, set rowcount on, and so on. Setting session options in any stored procedure has an effect only inside that stored procedure.
- You cannot create #temp tables to use later in the session. Once the procedure completes, the #temp tables drop away automatically and the original session settings are restored, as in any other stored procedure.
- You should 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.
- 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 must not 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 minus number, the login fails.

Acquiring the permissions of another user

Adaptive Server provides two ways of acquiring 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 481.
- **proxy authorization** allows one user to assume the identity of another user on a server-wide basis. See "Using proxy authorization" on page 483.

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     /*re-establishes original identity*/
```

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. A user with this permission can even assume the identity of the "sa" login, and, thereby, have unlimited power within Adaptive Server.

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.

Granting proxy authorization

System Security Officers use the grant set proxy or grant set session authorization command to give a user permission to impersonate another user within the server. The user with this permission can then execute either set proxy or set session authorization to become another user.

To grant proxy authorization permission, you must be a System Security Officer and execute the grant command from the master database. The syntax is:

```
grant set proxy
    to {public | name_list | role_name}
```

or

```
grant set session authorization
    to {public | name_list | role_name}
```

where:

- *public* – is all users. Sybase recommends that you not grant this permission to "public."
- *role_name* – is an Adaptive Server system or user-defined role. You can grant permissions to users based on the specific role granted.
- *name_list* – is user database or group names, separated by commas. The user must be a valid user in the master database.

To grant set proxy to an application with the login "appl" if you do not have sso_role currently active, and you are not in the master database, execute:

```
use master
go
set role sso_role on
go
grant set proxy to appl
go
```

To grant set proxy to that user-defined role "accountant," execute:

```
grant set proxy to accountant
```

To grant set session authorization to the "sa" account, whose user name in every database is "dbo," execute:

```
grant set proxy to dbo
```

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 that you are allowed to use. However, the *login_name* you specify must be a valid user in the database, or the database must have a "guest" user defined for it.
- Only one level is permitted; to impersonate more than one user, you must return to your original identity between impersonations.

- If you execute set proxy or set session authorization from within a procedure, your original identity is automatically resumed when you exit the procedure.

If you have a login that has been granted permission to use set proxy or set session authorization, you can set proxy to impersonate another user. The following is the syntax, where *login_name* is the name of a valid login in master..syslogins:

set proxy *login_name*

or

set session authorization *login_name*

Enclose the login name in quotation marks.

For example, to set proxy to “mary,” execute:

```
set proxy "mary"
```

After setting proxy, check your login name in the server and your user name in the database. For example, assume that your login is “ralph” and that you have been granted set proxy authorization. You want to execute some commands as “sallyn” and as “rudolph” in pubs2 database. “sallyn” has a valid name (“sally”) in the database, but Ralph and Rudolph do not. However, pubs2 has a guest user defined. You can execute:

```
set proxy "sallyn"
go
use pubs2
go
select suser_name(), user_name()
go
------------------------------ -------------------
sallyn                         sally
```

To change to Rudolph, you must first change back to your own identity. To do so, execute:

```
set proxy "ralph"
select suser_name(), user_name()
go
------------------------------ --------------------
ralph                          guest
```

Notice that Ralph is a “guest” in the database.

Then execute:

```
set proxy "rudolph"
```

```
go
select suser_name(), user_name()
go
------------------------------ --------------------
rudolph                        guest
```

Rudolph is also a guest in the database because Rudolph is not a valid user in the database.

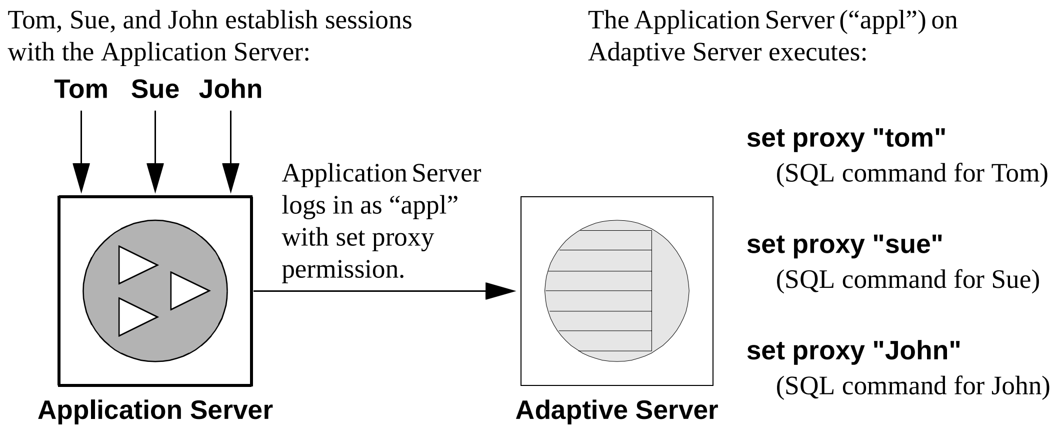
Now, impersonate the “sa” account. Execute:

```
set proxy "ralph"
go
set proxy "sa"
go
select suser_name(), user_name()
go
--------------------------- --------------------
sa                          dbo
```

Proxy authorization for applications

Figure 12-1 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 12-1: Applications and proxy authorization


Tom, Sue, and John establish sessions with the Application Server:
The Application Server (“appl”) on Adaptive Server executes:
Tom
Sue
John
Application Server logs in as “appl” with set proxy permission.
set proxy "tom"
(SQL command for Tom)
set proxy "sue"
(SQL command for Sue)
set proxy "John"
(SQL command for John)
Application Server
Adaptive Server

Reporting on permissions

Table 12-6 lists the system procedures for reporting information about proxies, object creation and object access permissions:

Table 12-6: System procedures for reporting on permissions

To report information on	Use
Proxies	system tables
Users and processes	sp_who
Permissions on database objects or users	sp_helprotect
Permissions on specific tables	sp_table_privileges
Permissions on specific columns in a table	sp_column_privileges

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
- 2 for revoke

For more information about the sysprotects table, see the *Reference Manual*.

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:

```
set proxy susie
```

sp_who returns "susie" for loginame and "ralph" for origname.

sp_who queries the master..sysprocesses system table, which contains columns for the server user ID (suid) and the original server user ID (origsuid).

For more information, see sp_who in the *Reference Manual*.

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:

```
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 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
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   titles     All         FALSE
dbo     judy   Grant   Update   titles     advance     FALSE
dbo     judy   Grant   Update   titles     notes       FALSE
dbo     judy   Grant   Update   titles     price       FALSE
dbo     judy   Grant   Update   titles     pub_id      FALSE
dbo     judy   Grant   Update   titles     pubdate     FALSE
dbo     judy   Grant   Update   titles     title       FALSE
dbo     judy   Grant   Update   titles     title_id    FALSE
dbo     judy   Grant   Update   titles     total_sales FALSE
dbo     judy   Grant   Update   titles     type        FALSE
```

The first row shows that the Database Owner (“dbo”) gave Judy permission to select all columns of the titles table. The rest of the lines indicate that she can update only the columns listed in the display. Judy cannot give select or update permissions to any other user.

To see Judy’s permissions on the publishers table, enter:

```
sp_helprotect publishers, judy
```

In this display, the grantable column indicates TRUE, meaning that Judy can grant the permission to other users.

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy    Grant  Select  publishers  all         TRUE
```

Reporting permissions on specific tables

Use sp_table_privileges to return permissions information about a specified table. The syntax is:

```
sp_table_privileges table_name [, table_owner
    [, table_qualifier]]
```

where:

- *table_name* – is the name of the table. It is required.
- *table_owner* – can be used to specify the name of the table owner, if it is not “dbo” or the user executing sp_table_privileges.
- *table_qualifier* – is the name of the current database.

Use null for parameters that you want to skip.

For example, the following statement:

```
sp_table_privileges titles
```

returns information about all permissions granted on the titles table. For more information about the output of sp_table_privileges see the *Reference Manual*.

Reporting permissions on specific columns

Use sp_column_privileges to return information about permissions on columns in a table. The syntax is:

```
sp_column_privileges table_name [, table_owner
    [, table_qualifier [, column_name]]]
```

where:

- *table_name* – is the name of the table.
- *table_owner* – can be used to specify the name of the table owner, if it is not “dbo” or the user executing sp_column_privileges.
- *table_qualifier* – is the name of the current database.
- *column_name* – is the name of the column on which you want to see permissions information.

Use null for parameters that you want to skip.

For example, the following statement:

```
sp_column_privileges publishers, null, null, pub_id
```

returns information about the pub_id column of the publishers table. For more information about the output of sp_column_privileges, see the *Reference Manual*.

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 494.

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 to 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 would hide 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 could 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:

```
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 non-sales 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 non-sales users would have to 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.

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 proc_role system function within the procedure to guarantee that a procedure can be executed only by users who have a given role. proc_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 proc_role to see if the user has the System Administrator role:

```
create proc test_proc
as
if (proc_role("sa_role") = 0)
begin
    print "You don’t have the right role"
    return -1
end
else
    print "You have SA role"
    return 0
```

See “System Functions” in the *Reference Manual* for more information about proc_role.

Understanding ownership chains

Views can depend on other views and/or tables. Procedures can depend on other procedures, views, and/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.

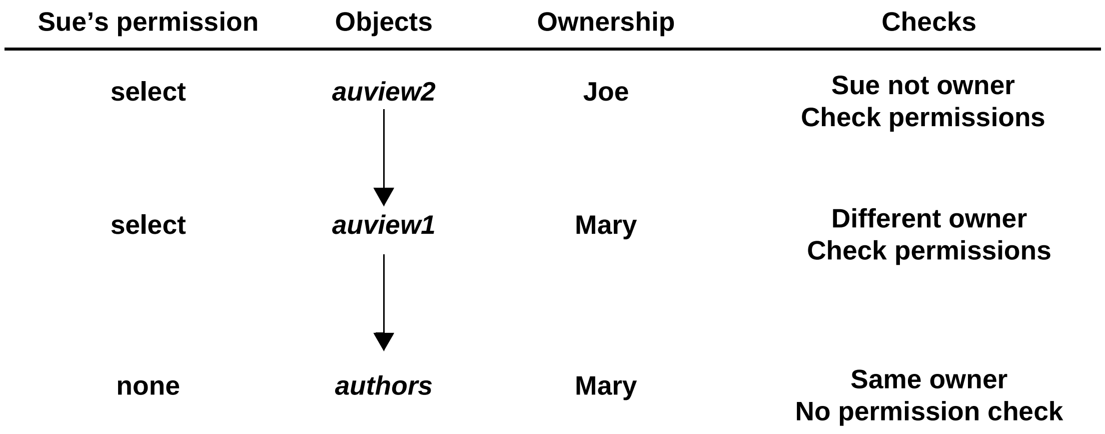
Ordinarily, a user who creates a view needs worry only about granting permissions on that view. For example, say Mary has created a view called auview1 on the authors table, which she also owns. If Mary grants select permission to Sue on auview1, Adaptive Server will let Sue access it without checking 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 12-2: Ownership chains and permission checking for views, case 1

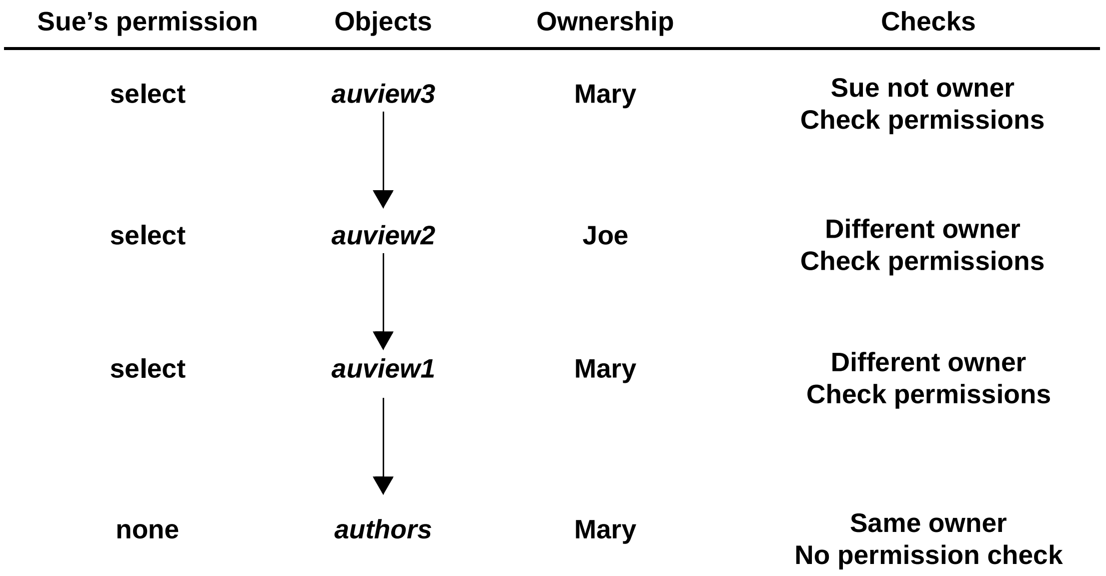


Adaptive Server checks the permissions on auview2 and auview1, and finds that Sue can use them. Adaptive Server checks ownership on auview1 and authors and finds that they have the same owner. Therefore, Sue can use auview2.

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:

Figure 12-3: Ownership chains and permission checking for views, case 2



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:

Figure 12-4: Ownership chains and permission checking for stored procedures

Sue's permission	Objects	Ownership	Checks
execute	***proc4***	**Mary**	**Sue not owner Check permissions**
none	***proc3***	**Mary**	**Same owner No permissions check**
execute	***proc2***	**Joe**	**Different owner Check permissions**
execute	***proc1***	**Mary**	**Different owner Check permissions**
none	***authors***	**Mary**	**Same owner No permission check**

To execute proc4, Sue must have permission to execute proc4, proc2, and proc1. Permission to execute proc3 is not necessary because proc3 and proc4 have the same owner.

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.

For more information on triggers, see the *Transact-SQL User's Guide* or the *Reference Manual*.

CHAPTER 13 Managing Remote Servers

This chapter discusses the steps the System Administrator and System Security Officer of each Adaptive Server must execute to enable **remote procedure calls** (RPCs).

Overview

Users on a local Adaptive Server can execute stored procedures on a remote Adaptive Server. Executing an RPC sends the results of the remote process to the calling process—usually displayed on the user's screen.

Note The use of remote servers is not included in the evaluated configuration.

To enable RPCs, the System Administrator and System Security Officer of each Adaptive Server must execute the following steps:

- On the local server:
  - *System Security Officer* – use sp_addserver to list the local server and remote server in the system table master..sysservers.
  - List the remote server in the *interfaces* file or Directory Service for the local server.

    - Restart the local server so the global variable *@@servername* is set to the name of the local server. If this variable is not set properly, users cannot execute RPCs from the local server on any remote server.

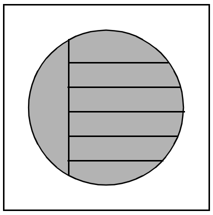
- On the remote server:
    - *System Security Officer* – use sp_addserver to list the server originating the RPC in the system table master..sysservers.
    - To allow the user who is originating the remote procedure access to the server, a System Security Officer uses sp_addlogin, and a System Administrator uses sp_addremotelogin.
    - Add the remote login name as a user of the appropriate database and grant that login permission to execute the procedure. (If execute permission is granted to "public", the user does not need to be granted specific permission.)

Figure 13-1 shows how to set up servers for remote access.

Figure 13-1: Setting up servers to allow remote procedure calls

The user "joe" on ROSE needs to access stored procedures on ZINNIA

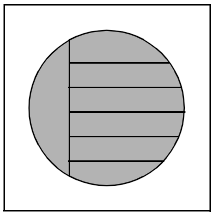
ROSE



sp_addserver ROSE, local
sp_addserver ZINNIA

Interfaces files must have an entry
for ZINNIA

ZINNIA



sp_addserver ROSE
sp_addlogin joe
sp_addremotelogin ROSE, joe

sp_adduser joe (in the appropriate database)
grant execute on *procedure_name* to joe

For operating-system-specific information about handling remote servers, see the the installation documentation for your platform.

Managing remote servers

Table 13-1 lists the tasks related to managing remote servers and the system procedures you use to perform the tasks.

Table 13-1: Tasks related to managing remote servers

To	Use	See
Add a remote server	sp_addserver	"Adding a remote server" on page 503
Manage remote server names	sp_addserver	"Managing remote server names" on page 504
Change server connection options	sp_serveroption	"Setting server connection options" on page 505
Display information about servers	sp_helpserver	"Getting information about servers" on page 507
Drop a server	sp_dropserver	"Dropping remote servers" on page 507

Adding a remote server

A System Security Officer uses sp_addserver to add entries to the sysservers table. On the server originating the call, you must add one entry for the local server, and one for each remote server that your server will call.

When you create entries for a remote server, you can either:

- Refer to them by the name listed in the *interfaces* file, or
- Provide a local name for the remote server. For example, if the name in the *interfaces* file is "MAIN_PRODUCTION," you may want to call it simply "main."

The syntax is:

```
sp_addserver lname [{, local | null}
    [, pname]]
```

where:

- *lname* – provides the local "call name" for the remote server. If this name is *not* the same as the remote server's name in the *interfaces* file, you must provide that name as the third parameter, *pname*.

  The remote server must be listed in the *interfaces* file on the local machine. If it's not listed, copy the *interfaces* file entry from the remote server and append it to your existing *interfaces* file. Be sure to keep the same port numbers.

- local – identifies the server being added as a local server. The local value is used only after start-up, or after a restart, to identify the local server name so that it can appear in messages printed out by Adaptive Server. null specifies that this server is a remote server.

  Note For users to be able to run RPCs successfully from the local server, the local server must be added with the local option and restarted. The restarting is required to set the global variable @@*servername*.

- *pname* – is the remote server listed in the *interfaces* file for the server named *lname*. This optional argument permits you to establish local aliases for any other Adaptive Server, Open Server™, or Backup Server that you may need to communicate with. If you do not specify *pname*, it defaults to *lname*.

Examples of adding remote servers

This example creates an entry for the local server named DOCS:

```
sp_addserver DOCS, local
```

This example creates an entry for a remote server named GATEWAY:

```
sp_addserver GATEWAY
```

To run a remote procedure such as sp_who on the GATEWAY server, execute either:

```
GATEWAY.sybsytemprocs.dbo.sp_who
```

or:

```
GATEWAY...sp_who
```

This example gives a remote server called MAIN_PRODUCTION the local alias “main:”

```
sp_addserver main, null, MAIN_PRODUCTION
```

The user can then enter:

```
main...sp_who
```

Managing remote server names

The master.dbo.sysservers table has two name columns:

- srvname is the unique server name that users must supply when executing remote procedure calls.
- srvnetname is the server's network name, which must match the name in the *interfaces* file.

To add or drop servers from your network, you can use sp_addserver to update the server's network name in srvnetname.

For example, to remove the server MAIN from the network, and move your remote applications to TEMP, you can use the following statement to change the network name, while keeping the local alias:

```
sp_addserver MAIN, null, TEMP
```

sp_addserver displays a message telling you that it is changing the network name of an existing server entry.

Setting server connection options

sp_serveroption sets the server options timeouts, net password encryption, rpc security model A, and rpc security model B, which affect connections with remote servers. Additionally, if you have set the remote procedure security model to rpc security model B, you can use sp_serveroption to set these additional options: security mechanism, mutual authentication, use message confidentiality, and use message integrity.

The options you specify for sp_serveroption do not affect the communication between Adaptive Server and Backup Server.

The following sections describe timeouts, net password encryption, rpc security model A, and rpc security model B. For information about the additional options you can specify when rpc security model B is on, see "Establishing security for remote procedures" on page 536.

Using the *timeouts* option

A System Administrator can use the timeouts option to disable and enable the normal timeout code used by the local server.

By default, timeouts is set to true, and the site handler process that manages remote logins times out if there has been no remote user activity for one minute. By setting timeouts to false on both of the servers involved in remote procedure calls, the automatic timeout is disabled. This example changes timeouts to false:

```
sp_serveroption GATEWAY, "timeouts", false
```

After you set timeouts to false on both servers, when a user executes an RPC in either direction, the site handler on each machine runs until one of the servers is shut down. When the server is brought up again, the option remains false, and the site handler will be reestablished the next time a user executes an RPC. If users execute RPCs frequently, it is probably efficient in terms of system resources to set this option to false, since there is some system overhead involved in setting up the physical connection.

Using the *net password encryption* option

A System Security Officer can use net password encryption to specify whether connections with a remote server are to be initiated with a client-side password encryption handshake or with the usual unencrypted password handshake sequence. The default is false.

If net password encryption is set to true:

1 The initial login packet is sent without passwords.

2 The client indicates to the remote server that encryption is desired.

3 The remote server sends back an encryption key, which the client uses to encrypt its plain text passwords.

4 The client then encrypts its own passwords, and the remote server uses the key to authenticate them when they arrive.

This example sets net password encription to true:

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

This option does not affect Adaptive Server’s interaction with Backup Server.

Using the *rpc security model* options

The rpc security model A and rpc security model B options determine what kind of security is available for RPCs. If you use model A, which is the default, Adaptive Server does not support security services such as message confidentiality via encryption between the two servers.

For security model B, the local Adaptive Server gets a credential from the security mechanism and uses the credential to establish a secure physical connection with the remote Adaptive Server. With this model, you can choose one or more of these security services: mutual authentication, message confidentiality via encryption, and message integrity.

To set security model A for the server GATEWAY, execute:

```
sp_serveroption GATEWAY, "rpc security model A",
   true
```

For information about how to set up servers for Security Model B, see “Establishing security for remote procedures” on page 536.

Getting information about servers

sp_helpserver reports on servers. Without an argument, it provides information about all the servers listed in sysservers. When you include a server name, it provides information about that server only. The syntax is:

sp_helpserver [*server*]

sp_helpserver checks for both srvname and srvnetname in the master..sysremotelogins table.

For operating-system-specific information about setting up remote servers, see the the installation documentation for your platform.

Dropping remote servers

A System Security Officer can use the sp_dropserver system procedure to drop servers from sysservers. The syntax is:

sp_dropserver *server* [, droplogins]

where:

- *server* – is the name of the server you want to drop.
- droplogins – allows you to drop a remote server and all of that server’s remote login information in one step. If you do not use droplogins, you cannot drop a server that has remote logins associated with it.

The following statement drops the GATEWAY server and all of the remote logins associated with it:

```
sp_dropserver GATEWAY, droplogins
```

You don't have to use droplogins if you want to drop the local server; that entry does not have remote login information associated with it.

Adding remote logins

The System Security Officer and System Administrator of any Adaptive Server share control over which remote users can access the server, and what identity the remote users assume. The System Administrator uses sp_addremotelogin to add remote logins and sp_dropremotelogin to drop remote logins. The System Security Officer uses sp_remoteoption to control whether password checking will be required.

Mapping users' server IDs

Logins from a remote server can be mapped to a local server in three ways:

- A particular remote login can be mapped to a particular local login name. For example, user "joe" on the remote server might be mapped to "joesmith".
- All logins from one remote server can be mapped to one local name. For example, all users sending remote procedure calls from the MAIN server might be mapped to "remusers".
- All logins from one remote server can use their remote names.

The first option can be combined with the other two options, and its specific mapping takes precedence over the other two more general mappings. The second and third options are mutually exclusive; you can use either of them, but not both.

Changing the mapping option

Use sp_dropremotelogin to remove the old mapping.

Use sp_addremotelogin to add remote logins. The syntax is:

```
sp_addremotelogin remoteserver [, loginame
    [, remotename]]
```

If the local names are not listed in master..syslogins, add them as Adaptive Server logins with sp_addlogin before adding the remote logins.

Only a System Administrator can execute sp_addremotelogin. For more information, see the *Reference Manual*.

Mapping remote logins to particular local names

The following example maps the login named “pogo” from a remote system to the local login name “bob”. The user logs in to the remote system as “pogo”. When that user executes remote procedure calls from GATEWAY, the local system maps the remote login name to “bob”.

```
sp_addlogin bob
sp_addremotelogin GATEWAY, bob, pogo
```

Mapping all remote logins to one local name

The following example creates an entry that maps all remote login names to the local name “albert”. All names are mapped to “albert”, except those with specific mappings, as described in the previous section. For example, if you mapped “pogo” to “bob”, and then the rest of the logins to “albert”, “pogo” still maps to “bob”.

```
sp_addlogin albert
sp_addremotelogin GATEWAY, albert
```

If you use sp_addremotelogin to map all users from a remote server to the same local name, use sp_remoteoption to specify the “trusted” option for those users. For example, if all users from server GATEWAY that are mapped to “albert” are to be trusted, specify:

```
sp_remoteoption GATEWAY, albert, NULL, trusted, true
```

If you do not specify the logins as trusted, the logins will not be allowed to execute RPCs on the local server unless they specify passwords for the local server when they log in to the remote server. Users, when they use Open Client Client-Library can use the routine ct_remote_pwd to specify a password for server-to-server connections. isql and bcp do not permit users to specify a password for RPC connections. See “Password checking for remote users” on page 512 for more information about sp_remoteoption.

Warning! Do not map more than one remote login to a single local login, as it reduces individual accountability on the server. Audited actions can be traced only to the local server login, not to the individual logins on the remote server.

If you are using network based security

If users are logged into the remote server using “unified login”, the logins must also be trusted on the local server, or they must specify passwords for the server when they log into the remote server. For information about “unified login”, see “Using unified login” on page 529.

Warning! Using the trusted mode of sp_remoteoption reduces the security of your server, as passwords from such “trusted” users are not verified.

Keeping remote login names for local servers

To enable remote users to keep their remote login names while using a local server:

1. Use sp_addlogin to create a login for each login from the remote server.
2. Use sp_addremotelogin for the server as a whole to create an entry in master..sysremotelogins with a null value for the remote login name and a value of -1 for the suid. For example:

   ```
   sp_addremotelogin GATEWAY
   ```

Example of remote user login mapping

This statement displays the local and remote server information recorded in master..sysservers:

```
select srvid, srvname from sysservers
srvid  srvname
-----  ----------
    0  SALES
    1  CORPORATE
    2  MARKETING
    3  PUBLICATIONS
    4  ENGINEERING
```

The SALES server is local. The other servers are remote.

This statement displays information about the remote servers and users stored in master..sysremotelogins:

```
select remoteserverid, remoteusername, suid
  from sysremotelogins
```

```
remoteserverid   remoteusername   suid
--------------   --------------   ------
1                joe              1
1                nancy            2
1                NULL             3
3                NULL             4
4                NULL            -1
```

By matching the value of remoteserverid in this result and the value of srvid in the previous result, you can find the name of the server for which the remoteusername is valid. For example, in the first result, srvid 1 indicates the CORPORATE server; in the second result remoteserverid 1 indicates that same server. Therefore, the remote user login names "joe" and "nancy" are valid on the CORPORATE server.

The following statement shows the entries in master..syslogins:

```
select suid, name from syslogins
suid    name
------   ------------
    1   sa
    2   vp
    3   admin
    4   writer
```

The results of all three queries together show:

- The remote user name "joe" (suid 1) on the remote CORPORATE server (srvid and remoteserverid 1) is mapped to the "sa" login (suid 1).
- The remote user name "nancy" (suid 2) on the remote CORPORATE server (srvid and remoteserverid 1) is mapped to the "vp" login (suid 2).
- The other logins from the CORPORATE server (remoteusername "NULL") are mapped to the "admin" login (suid 3).
- All logins from the PUBLICATIONS server (srvid and remoteserverid 3) are mapped to the "writer" login (suid 4).
- All logins from the ENGINEERING server (srvid and remoteserverid 4) are looked up in master..syslogins by their remote user names (suid -1).
- There is no remoteserverid entry for the MARKETING server in sysremotelogins. Therefore, users who log in to the MARKETING server cannot run remote procedure calls from that server.

The remote user mapping procedures and the ability to set permissions for individual stored procedures give you control over which remote users can access local procedures. For example, you can allow the “vp” login from the CORPORATE server to execute certain local procedures and all other logins from CORPORATE to execute the procedures for which the “admin” login has permission.

Note In many cases, the passwords for users on the remote server must match passwords on the local server.

Password checking for remote users

A System Security Officer can use sp_remoteoption to determine whether passwords will be checked when remote users log in to the local server. By default, passwords are verified (“untrusted” mode). In trusted mode, the local server accepts remote logins from other servers and front-end applications without user-access verification for the particular login.

When sp_remoteoption is used with arguments, it changes the mode for the named user. The syntax is:

sp_remoteoption [*remoteserver*, *loginame*, *remotename*,
    optname, {true | false}]

The following example sets trusted mode for the user “bob”:

```
sp_remoteoption GATEWAY, pogo, bob, trusted,
    true
```

Effects of using the untrusted mode

The effects of the “untrusted” mode depend on the user’s client program. isql and some user applications require that logins have the same password on the remote server and the local server. Open Client™ applications can be written to allow local logins to have different passwords on different servers.

To change your password in "untrusted" mode, you must first change it on all the remote systems you access before changing it on your local server. This is because of the password checking. If you change your password on the local server first, when you issue the remote procedure call to execute sp_password on the remote server your passwords will no longer match.

The syntax for changing your password on the remote server is:

remote_server...sp_password *caller_passwd*, *new_passwd*

On the local server, the syntax is:

sp_password *caller_passwd*, *new_passwd*

See "Changing passwords" on page 368 for more information about changing your password.

Getting information about remote logins

sp_helpremotelogin prints information about the remote logins on a server. The following example shows the remote login "pogo" mapped locally to login name "bob", with all other remote logins keeping their remote names.

```
sp_helpremotelogin
```

```
server     remote_user_name     local_user_name      options
---------    ----------------     -----------------    --------
GATEWAY    **mapped locally**   **use local name**    untrusted
GATEWAY    pogo                 bob                   untrusted
```

Configuration parameters for remote logins

Table 13-2 shows the configuration parameters that affect RPCs. All these configuration parameters are set using sp_configure and do not take effect until Adaptive Server is restarted.

Table 13-2: Configuration parameters that affect RPCs

Configuration parameter	Default
allow remote access	1
number of remote logins	20
number of remote sites	10
number of remote connections	20
remote server pre-read packets	3

Allowing remote access

To allow remote access to or from a server, including Backup Server, set allow remote access to 1:

```
sp_configure "allow remote access", 1
```

To disallow remote access at any time, set allow remote access to 0:

```
sp_configure "allow remote access", 0
```

Only a System Security Officer can set the allow remote access parameter.

Note You cannot perform database or transaction log dumps while the allow remote access parameter is set to 0.

Controlling the number of active user connections

To set the number of active user connections from this site to remote servers, use number of remote logins. This command sets number of remote logins to 50:

```
sp_configure "number of remote logins", 50
```

Only a System Administrator can set the number of remote logins parameter.

Controlling the number of remote sites

To control the number of remote sites that can access a server simultaneously, use number of remote sites. All accesses from an individual site are managed by one site handler. This parameter controls the number of site handlers, not the number of individual, simultaneous procedure calls. For example, if you set number of remote sites to 5, and each site initiates three remote procedure calls, sp_who shows 5 site handler processes for the 15 processes. Only a System Administrator can set the number of remote sites.

Controlling the number of active remote connections

To control the limit on active remote connections that are initiated to and from a server, use the number of remote connections parameter. This parameter controls connections initiated from the server and connections initiated from remote sites to the server. Only a System Administrator can set number of remote connections.

Controlling number of preread packets

To reduce the needed number of connections, all communication between two servers is handled through one site handler. This site handler can preread and keep track of data packets for each user before the user process that needs them is ready.

To control how many packets a site handler will preread, use remote server pre-read packets. The default value, 3, is adequate in all cases; higher values can use too much memory. Only a System Administrator can set remote server pre-read packets. For more information, see “remote server pre-read packets” on page 130.

CHAPTER 14

Using Kerberos, DCE, and Windows NT LAN Manager

This chapter describes the network-based security services that enable you to authenticate users and protect data transmitted among machines on a network.

For information about the Secure Sockets Layer (SSL) security mechanism, see Chapter 9, “Security Administration.”

Overview

In a distributed client/server computing environment intruders can view or tamper with confidential data. Adaptive Server works with third-party providers to give you security services that:

- Authenticate users, clients, and servers – make sure they are who they say they are.
- Provide data confidentiality with encryption – ensure that data cannot be read by an intruder.
- Provide data integrity – prevent data tampering and detect when it has occurred.

Table 14-1 lists the security mechanisms supported by Adaptive Server on UNIX and desktop platforms.

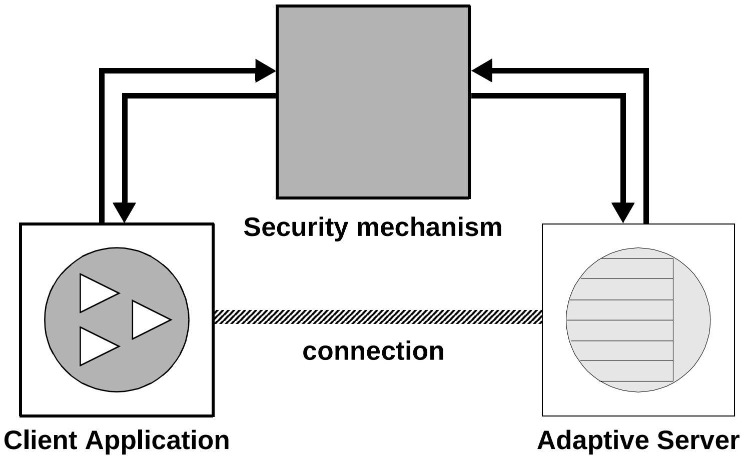
Table 14-1: Security mechanisms supported by Adaptive Server

Security mechanism	Supported platforms
Distributed Computing Environment (DCE)	Solaris 32bit & AIX 32bit
CyberSAFE Kerberos	Solaris 32bit, AIX 32bit, Windows NT
Windows NT LAN Manager	Windows NT

How applications use security services

The following illustration shows a client application using a security mechanism to ensure a secure connection with Adaptive Server.

Figure 14-1: Establishing secure connections between a client and Adaptive Server



The secure connection between a client and a server can be used for:

- Login authentication
- Message protection

Login authentication

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.

Message protection

If the client requests message protection services:

1. The client uses the security mechanism to prepare the data packet it will send 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. When Adaptive Server receives the data packet, it 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 upon 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. This must be requested by the client and 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.
- Remote procedure security – establishes mutual authentication, message confidentiality, and message integrity for remote procedure communications.

Note The security mechanism you are using may not all of these services. For information about what services are available to you, see "Getting information about available security services" on page 547.

Administering network-based security

Table 14-2 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 14-2.

Table 14-2: Process for administering network-based security

Step	Description	See
1. Set up the configuration files: • *libtcl.cfg* • *objectid.dat* • *interfaces* (or Directory Service)	Edit the *libtcl.cfg* file. Edit the *objectid.dat* file. Edit the *interfaces* file or Directory Service.	• "Setting up configuration files for security" on page 521 • The *Open Client/Server Configuration Guide* for your platform.
2. Make sure the security administrator for the security mechanism has created logins for each user and for the Adaptive Server and Backup Server.	The security administrator must add names and passwords for users and servers in the security mechanism. For DCE, the security administrator needs to create a *keytab* file for server entries.	• The documentation supplied with your security mechanism. • "Identifying users and servers to the security mechanism" on page 527.
3. Configure security for your installation.	Use sp_configure.	"Configuring Adaptive Server for security" on page 528.
4. Restart Adaptive Server.	Activates the use security services parameter.	"Restarting the server to activate security services" on page 532.

Step	Description	See
5. Add logins to Adaptive Server to support enterprise-wide login.	Use sp_addlogin to add users. Optionally, specify a default secure login with sp_configure.	"Adding logins to support unified login" on page 535.
6. Determine the security model for remote procedures and set up the local and remote servers for RPC security.	Use sp_serveroption to choose the security model (A or B).	"Establishing security for remote procedures" on page 536.
7. Connect to the server and use security services.	Use isql_dce or isql_r (if you are using DCE library services or security services) 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 544. • The *Open Client/Server Configuration Guide* for your platform. • "Security Features" topics page in the *Open Client Client-Library/C Reference Manual.*
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 547.

Setting up configuration files for security

Configuration files are created during installation at a default location in the Sybase directory structure. Table 14-3 provides an overview of the configuration files required for using network-based security.

Table 14-3: Names and locations for configuration files

File name	Description	Location
libtcl.cfg	The driver configuration file contains information regarding directory, security, and network drivers and any required initialization information.	*UNIX platforms*: *$SYBASE/config* *Desktop platforms*: *SYBASE_home\ini*
objectid.dat	The object identifiers file maps global object identifiers to local names for character set, collating sequence, and security mechanisms.	*UNIX platforms*: *$SYBASE/config* *Desktop platforms*: *SYBASE_home\ini*

File name	Description	Location
UNIX: *interfaces* *Desktop platforms*: *sql.ini*	The *interfaces* file contains connection and security information for each server listed in the file. **Note** In this version, you can use a Directory Service instead of the *interfaces* file.	*UNIX platforms*: *$SYBASE* *Desktop platforms*: *SYBASE_home\ini*

For a detailed description of the configuration files, see the *Open Client/Server Configuration Guide* for your platform.

Preparing*libtcl.cfg* to use network-based security

libtcl.cfg contains information about three types of drivers:

- Network (Net-Library)
- Directory Services
- Security

A **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 re-linking the application.

Entries for network drivers

The syntax for a network driver entry is:

driver=protocol description

where:

- *driver* – is the name of the network driver.
- *protocol* – is the name of the network protocol.

- *description* – is a description of the entry. This element is optional.

> **Note** If you do not specify a network driver, an appropriate driver for your application and platform is automatically used. For example, for UNIX platforms, a driver that can handle threads is automatically chosen when security services are being used.

Entries for Directory Services

Entries for Directory Services apply if you want to use a Directory Service instead of the *interfaces* file. For information about directory entries, see the configuration documentation for your platform, and the *Open Client/Server Configuration Guide* for your platform.

Entries for security drivers

The syntax for a security driver entry is:

provider=driver init-string

where:

- *provider* – is the local name for the security mechanism. The mapping of the local name to a global object identifier is defined in *objectid.dat*.

  The default local names are:

  - "dce" – For the DCE security mechanism.
  - "csfkrb5" – For the CyberSAFE Kerberos security mechanism.
  - "LIBSMSSP" – For Windows LAN Manager on Windows NT or Windows 95 (clients only).

  If you use a local mechanism name other than the default, you must change the local name in the objectid.dat file (see "The objectid.dat file" on page 525 for an example).

- *driver* – is the name of the security driver. The default location of all drivers for UNIX platforms is *$SYBASE/lib*. The default location for desktop platforms is SYBASE_home\dll.

- *init-string* – is an initialization string for the driver. This element is optional. The value for *init-string* varies by driver:

  - DCE driver – the following is the syntax for *init-string*, where *cell_name* is the name of your DCE cell:

secbase=/.../*cell_name*

- CyberSAFE Kerberos driver – the following is the syntax for *init-string*, where *realm* is the default CyberSAFE Kerberos realm name:

  secbase=@*realm*

- Windows NT LAN Manager – *init-string* is not applicable.

UNIX platform information

This section contains information specific to UNIX platforms. For more information, see the *Open Client/Server Configuration Guide for UNIX*.

UNIX platforms – 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.

The *libtcl.cfg* file, after installation of Adaptive Server on a UNIX platform, 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 “;”).

Sample *libtcl.cfg* File 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.

[DIRECTORY]
;dce=libddce.so ditbase=/.:/subsys/sybase/dataservers
;dce=libddce.so ditbase=/.:/users/cfrank

[SECURITY]
dce=libsdce.so secbase=/.../svrsole4_cell
```

This *libtcl.cfg* file is set up to use the DCE security service. Notice that 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 chosen automatically by the system. A threaded driver is chosen automatically when security services are being used, and a non-threaded driver is chosen automatically for applications that cannot work with threaded drivers. For example, Backup Server does not support security services and does not work with a threaded driver.

Desktop platform information

This section contains information specific to desktop platforms. For more information, see the *Open Client/Server Configuration Guide for Desktop Platforms*.

Use the ocscfg utility to edit the *libtcl.cfg* file. See the *Open Client/Server Configuration Guide for Desktop Platforms* for instructions for using ocscfg.

The ocscfg utility creates section headings automatically for the *libtcl.cfg* file.

Sample *libtcl.cfg* file for desktop platforms

```
[NT_DIRECTORY]
ntreg_dsa=LIBDREG  ditbase=software\sybase\serverdsa

[DRIVERS]
NLWNSCK=TCP  Winsock TCP/IP Net-Lib driver
NLMSNMP=NAMEPIPE  Named Pipe Net-Lib driver
NLNWLINK=SPX  NT NWLINK SPX/IPX Net-Lib driver
NLDECNET=DECNET  DecNET Net-Lib driver

[SECURITY]
NTLM=LIBSMSSP
```

The *objectid.dat* file

The objectid.dat file maps global object identifiers, such as the one for the DCE service (“1.3.6.1.4.1.897.4.6.1”) to local names, such as “dce”. The file contains sections such as [CHARSET] for character sets and [SECURITY] for security services. Of interest here is the security section. Following is a sample objectid.dat file:

```
[secmech]
       1.3.6.1.4.1.897.4.6.1   = dce
       1.3.6.1.4.1.897.4.6.3   = NTLM
       1.3.6.1.4.1.897.4.6.6   = csfkrb5
```

You need to change this file only if you have changed the local name of a security service in the *libtcl.cfg* file. Use a text editor to edit the file.

For example, if you changed

```
[SECURITY]
dce=libsdce.so secbase=/.../svrsole4_cell
```

to

```
[SECURITY]
dce_group=libsdce.so secbase=/.../svrsole4_cell
```

in *libtcl.cfg*, then you need to change the objectid.dat file to reflect the change. Simply change the local name in the line for DCE in objectid.dat:

```
1.3.6.1.4.1.897.4.6.1    = dce_group
```

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

Specifying security information for the server

You can choose to 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. If you plan to use security services, the *interfaces* file must include a “secmech” line, which gives the global identifier or identifiers of the security services you plan to use.

Instead of using the *interfaces* file, 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. If you plan to use security services with a Directory Service, the secmech security attribute must be defined. It must point to one or more global identifiers of the security services you plan to use.

UNIX tools for specifying the security mechanism

To specify the security mechanism or mechanisms you want to use:

- If you are using the *interfaces* file, use the dscp utility.

- If you are using a Directory Service, use the dscp_r or dscp_dce 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*.

Desktop tools for specifying server attributes

To provide information about the servers for your installation in the sql.ini file or a Directory Service, use the dsedit utility. This utility provides a graphical user interface for specifying server attributes such as the server version, name, and security mechanism. For the security mechanism attribute, you can specify one or more object identifiers for the security mechanisms you plan to use. For information about using dsedit, see the *Open Client/Server Configuration Guide for Desktop Platforms*.

Identifying users and servers to the security mechanism

The security administrator for the security mechanism must define *principals*, which include both users and servers, to the security mechanism. Table 14-4 lists tools you can use to add users and servers.

Table 14-4: Defining users and servers to the security mechanism

Security mechanism	Command or tool
DCE	Use the DCE dcecp tool's user create command to create a new principal (user or server). In addition, use the keytab create command to create a DCE keytab file, which contains a principal's password in encrypted form. When you are defining a server to DCE, use command options that specify that the new principal can act as a server.
CyberSAFE Kerberos	Use the CyberSAFE kadmin utility's add command. In addition, use the kadmin utility, with the ext command to create a key in a CyberSAFE Kerberos server key table file. When you are defining a server to CyberSAFE Kerberos, use command options that specify that the new principal can act as a server.

Security mechanism	Command or tool
Windows NT LAN Manager	Run the User Manager tool to define users to the Windows NT LAN Manager. Be sure to define the Adaptive Server name as a user to Windows NT LAN Manager and bring up Adaptive Server as that user name.

Note In a production environment, you must control the 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.

Refer to 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.

Configuration parameters are used to:

- Enable network-based security
- Require unified login
- Require message confidentiality with data encryption
- Require one or more message integrity security services

Enabling network-based security

To enable or disable network-based security, use sp_configure to set the use security services configuration parameter. Set this parameter to 1 to enable network-based security. If this parameter is 0 (the default), network-based security services are not available. The syntax is:

```
sp_configure "use security services", [0|1]
```

For example, to enable security services, execute:

```
sp_configure "use security services", 1
```

Note This configuration parameter is static; you must restart Adaptive Server for it to take effect. See "Restarting the server to activate security services" on page 532.

Using unified login

Configuration parameters are available to:

- Require unified login
- Establish a default secure login

All the parameters for unified login take effect immediately. You must be a System Security Officer to set the parameters.

Requiring unified login

To require all users, other than the user with System Security Officer (sso) role, 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 username 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, that user name is used by Adaptive Server. For example, if a user logs in to the DCE security mechanism as "ralph," and "ralph" is a name 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 into 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 pre-authenticated by a security mechanism. The syntax is:

sp_configure "secure default login", 0, *login_name*

The default value for secure default login is "guest."

This login must be a valid login in master..syslogins. For example, to set the login "gen_auth" to be the default login:

1. Use sp_addlogin to add the login as a valid user in Adaptive Server:

   ```
   sp_addlogin gen_auth, pwgenau
   ```

   This procedure sets the initial password to "pwgenau".

2. Use sp_configure to designate the login as the security default.

   ```
   sp_configure "secure default login", 0, gen_auth
   ```

   Adaptive Server will use this login for a user who is pre-authenticated 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 sp_addlogin to add all users to the server.

For more information about adding logins, see "Adding logins to support unified login" on page 535 and "Adding logins to Adaptive Server" on page 346.

Mapping security mechanism login names to server names

Some security mechanisms may allow login names that are not valid in Adaptive Server. For example, login names that are longer than 30 characters, or login names containing special characters such as !, %, *, and & are invalid names in Adaptive Server. All login names in Adaptive Server must be valid identifiers. For information about what identifiers are valid, see Chapter 3, "Expressions, Identifiers, and Wildcard Characters," in the *Reference Manual*.

Table 14-5 shows how Adaptive Server converts invalid characters in login names:

Table 14-5: Conversion of invalid characters in login names

Invalid characters	Converts to
Ampersand & Apostrophe ’ Backslash \ Colon : Comma , Equals sign = Left quote ‘ Percent % Right angle bracket > Right quote ’ Tilde ~	Underscore _
Caret ^ Curly braces { } Exclamation point ! Left angle bracket < Parenthesis ( ) Period . Question mark ?	Dollar sign $
Asterisk * Minus sign - Pipe | Plus sign + Quotation marks " Semicolon ; Slash / Square brackets [ ]	Pound sign #

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 for setting this parameter 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 following configuration parameters to require that one or more types of data integrity be checked for all messages:

- msg integrity reqd – set this parameter to 1 to require that all messages be checked for general tampering. If this parameter 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
```

For information about estimating and specifying memory requirements, see the Chapter 18, “Configuring Memory.”

Restarting the server to activate security services

Once you have configured security services, you must restart Adaptive Server.

Windows NT – see the configuration documentation for your platform.

UNIX platforms – note that:

- After you complete the installation of Adaptive Server, your runserver file contains an invocation of the dataserver utility to start Adaptive Server.

- Two versions of the dataserver utility are available: dataserver_dce, and dataserver. Likewise, two versions of the diagserver are available: diagserver_dce and diagserver. The utility you use depends on the platform you use:
  - For Sun Solaris platforms, use dataserver if you plan to use security services and dataserver if you do not plan to use security services.
  - For HP and RS/6000 platforms, use dataserver and diagserver. You can use a single binary, whether or not you are using security services.
- If you are using the DCE security service, be sure you have defined the *keytab* file. You can specify the -K option to dataserver_dce to specify the location of the *keytab* file. If you do not specify a location, Adaptive Server assumes the file is located in *$SYBASE/config/$DSLISTEN_key*. Optionally, you can specify the location as follows:

  ```
  $SYBASE/bin/dataserver_dce -Stest4 -dd_master
     -K/opt/dcelocal/keys/test4_key
  ```

  This dataserver_dce command starts the server using the master device d_master and the keytab file stored in */opt/dcelocal/keys/test4_key*.

  If you are using the default location for *keytab*, and $DSLISTEN is set to the name of your server (test4), you can execute:

  ```
  $SYBASE/bin/dataserver_dce -dd_master
  ```

  Then, Adaptive Server looks for the *keytab* file in *$SYBASE/config/test4_key*.

  For information about setting up your *keytab* file for DCE, refer to the DCE administrative documentation.

Determining security mechanisms to support

use security services is set to 0, Adaptive Server supports no security mechanisms.

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 546.

CyberSafe Kerberos 5

CyberSafe Kerberos 5 assumes the KDC is running and properly configured for your realm, and the SyberSafe Trustbroker client liberaries are installed under or on each client host in your realm. For configuration information, consult the CyberSafe documentation and the man pages that come with the CyberSafe Kerberos software.

Starting Adaptive Server under Kerberos

1 Add the Adaptive Server name to the KDC and extract the service key to a key table file. For example:

```
/krb5bin/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 kadmin to search the /krb5/v5srvtab file (specified using the -t option) for the admin and the Adaptive Server key, instead of prompting for a password. This is useful for writing shell scripts.

Adding logins to support unified login

When users log in to Adaptive Server with a pre-authenticated 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 as that login. If a default login is not defined, Adaptive Server rejects the login.

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 login with the default login. You must add the default login in master..syslogins and use sp_configure to define the default. For details, see "Establishing a secure default login" on page 529.

General procedure for adding logins

Follow the general procedure described in Table 14-6 to add logins to the server and, optionally, to add users to one or more databases with appropriate roles and authorizations to one or more databases.

Table 14-6: Adding logins and authorizing database access

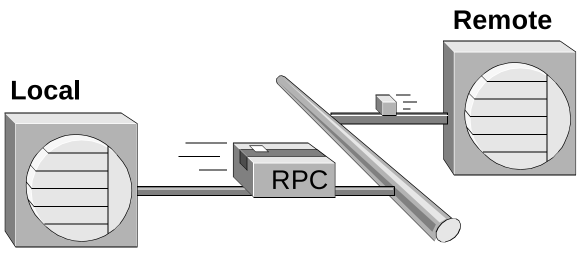
Task	Required role	Command or procedure	See
1. Add a login for the user.	System Security Officer	sp_addlogin	"Adding logins to Adaptive Server" on page 346

Task	Required role	Command or procedure	See
2. Add the user to one or more databases.	System Administrator or Database Owner	sp_adduser – execute this procedure from within the database.	"Adding users to databases" on page 349
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 370 • sp_changegroup in the *Reference Manual*
4. Grant system roles to the user.	System Administrator or System Security Officer	grant role	• "Creating and assigning roles to users" on page 356 • 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 356 in the *Reference Manual* • grant in the *Reference Manual* • create role in the *Reference Manual*
6. Grant access to database objects.	Database object owners		Chapter 12, "Managing User Permissions"

Establishing security for remote procedures

Adaptive Server acts as the client when it connects to another server to execute a remote procedure call (RPC) as shown in Figure 14-2.

Figure 14-2: Adaptive Server acting as client to execute an RPC



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 11.5 and later supports two security models for RPCs: *security model A* and *security model B.*

Security model A

For security model A, Adaptive Server does not support security services such as message confidentiality via encryption between the two servers. Security Model A is the default.

Security model B

For security model B, the local Adaptive Server gets a credential from the security mechanism and uses the credential to establish a secure physical connection with the remote Adaptive Server. With this model, you can use one or more of these security services:

- Mutual authentication – the local server authenticates the remote server by retrieving the credential of the remote server and verifying it with the security mechanism. With this service, the credentials of both servers are authenticated and verified.
- Message confidentiality via encryption – messages are encrypted when sent to the remote server, and results from the remote server are encrypted.
- Message integrity – messages between the servers are checked for tampering.

Unified login and the remote procedure models

If the local server and remote server are set up to use security services, you can use unified login on both servers with *either* model, using one of these two methods:

- The System Security Officer defines a user as "trusted" with sp_remoteoption on the remote server. With this method, a security mechanism such as DCE authenticates the user and password. The user gains access to the local server via "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. For more information about this routine, see the *Open Client Client-Library/C Reference Manual*.

Establishing the security model for RPCs

To establish the security model for RPCs, use sp_serveroption. The syntax is:

sp_serveroption *server*, *optname*, [true | false]

To establish the security model, set *optname* to rpc security model A or rpc security model B. *server* names the remote server.

For example, to set model B for remote server TEST3, execute:

```
sp_serveroption test3, "rpc security model B", true
```

The default model is "A," that is, remote procedure calls are handled the same as in previous versions. No server options need to be set for model A.

Setting server options for RPC security model B

For RPC security model B, you can set options with the sp_serveroption system procedure. The syntax is:

sp_serveroption *server*, *optname*, *optvalue*

where:

- *server* – is the name of the remote server.

- *optname* – is the name of the option. Values can be:
  - security mechanism – the name of the security mechanism to use when running an RPC on a remote server.
  - mutual authentication – set this option to 1 to cause the local Adaptive Server to authenticate and verify the remote server. If this parameter is 0 (the default), the remote server still verifies the local server when it sends an RPC, but the local server does not check the validity of the remote server.
  - use message confidentiality – set this option to 1 to cause all messages for the RPCs to be encrypted when they are sent to the remote server and received from the remote server. If this parameter is 0 (the default), data for the RPCs will not be encrypted.
  - use message integrity – set this option to 1 to require that all RPC messages be checked for tampering. If this parameter is 0 (the default), RPC data will not be checked for tampering.
- *optvalue* – must be equal to "true" or "false" for all values of *optname*, *except* security mechanism. If the option you are setting is security mechanism, specify the name of the security mechanism. To find the list of security mechanisms, execute:

  ```
  select * from syssecmechs
  ```

  For information about the syssecmechs system table, see "Determining enabled security services" on page 548.

For example, to set up the local server to execute RPCs on a remote server, TEST3, which will use the "dce" security mechanism, and to use mutual authentication for all RPCs between the two servers, execute:

```
sp_serveroption TEST3, "security mechanism", dce
sp_serveroption TEST3, "mutual authentication",
     true
```

Rules for setting up security model B for RPCs

Follow these rules when setting up security model B for RPCs:

- Both servers must be using security model B.
- Both servers must be using the same security mechanism, and that security mechanism must support the security services set with sp_serveroption.

- The System Security Officer of the local server must specify any security services that are required by the remote server. For example, if the remote server requires that all messages use the message confidentiality security service, the System Security Officer must use sp_serveroption to activate use message confidentiality.
- Logins who are authenticated by a security mechanism and log into Adaptive Server using “unified login” will not be permitted to execute RPCs on the remote procedure unless the logins are specified as “trusted” on the remote server or the login specifies the password for the remote server. Users, when they use Open Client Client-Library can use the routine ct_remote_pwd to specify a password for server-to-server connections. A System Administrator on Adaptive Server can use sp_remoteoption to specify that a user is trusted to use the remote server without specifying a password.

Preparing to use security model B for RPCs

Table 14-7 provides steps for using security model B to establish security for RPCS.

Table 14-7: Process for using security model B for RPCs

Task, who performs it, and where	Command, system procedure, or tool	See
System Administrator from the operating system: 1. Make sure the *interfaces* file or the Directory Service contains an entry for both servers and a secmech line listing the security mechanism.	UNIX: dscp or dscp_dce Desktop: dsedit	“Specifying security information for the server” on page 526 dscp or dscp_dce in the *Open Client/Server Configuration Guide for UNIX*. dsedit in the *Open Client/Server Configuration Guide for Desktop Platforms*.
System Security Officer on remote server: 2. Add the local server to master..sysservers.	sp_addserver Example: `sp_addserver "lcl_server"`	“Adding a remote server” on page 503. sp_addserver in the *Reference Manual*.
System Security Officer on remote server: 3. Add logins to master..syslogins.	sp_addlogin Example: `sp_addlogin user1, "pwuser1"`	“Adding logins to Adaptive Server” on page 346. sp_addlogin in the *Reference Manual*.

Task, who performs it, and where	Command, system procedure, or tool	See

System Security Officer on remote server:
4. Set use security services on, and set the rpc security model B as the model for connections with the local server.
	sp_configure – to set use security services
	sp_serveroption – to set the RPC security model
	Example:
	`sp_configure "use security services", 1`
	`sp_serveroption lcl_server, "rpc security model B", true`
		"Establishing the security model for RPCs" on page 538.
		"Enabling network-based security" on page 528.
		use security services (Windows NT only) in Chapter 4, "Setting Configuration Parameters" in this manual.
		sp_configure and sp_serveroption in the *Reference Manual*.

System Administrator on remote server: 5. Optionally, specify certain users as "trusted" to log into the remote server from the local server without supplying a password.	sp_remoteoption Example: `sp_remoteoption lcl_server, user1, user1, trusted, true`	"Password checking for remote users" on page 512. sp_remoteoption in the *Reference Manual*.

System Security Officer on local server:
6. Add both the local server and the remote server to master..sysservers.
	sp_addserver
	Example:
	`sp_addserver lcl_server, local`
	`sp_addserver rem_server`
		"Adding a remote server" on page 503.
		sp_addserver in the *Reference Manual*.

System Security Officer on local server: 7. Add logins to master..logins.	sp_addlogin Example:`sp_addlogin user1, "pwuser1"`	"Adding logins to Adaptive Server" on page 346. sp_addlogin in the *Reference Manual*.

System Security Officer on local server:
8. Set use security services on, and set the rpc security model B as the model for connections with the remote server.
	sp_configure – to set use security services
	sp_serveroption – to set the RPC security model
	Example:
	`sp_configure "use security services", 1`
	`sp_serveroption rem_server, "rpc security model B", true`
		"Establishing the security model for RPCs" on page 538.
		"Enabling network-based security" on page 528.
		use security services (Windows NT only) in Chapter 4, "Setting Configuration Parameters" in this manual.
		sp_configure and sp_serveroption in the *Reference Manual*.

Task, who performs it, and where	Command, system procedure, or tool	See

System Security Officer on local server:
9. Specify the security mechanism and the security services to use for connections with the remote server.
	sp_serveroption
	Example:
	`sp_serveroption rem_server, "security mechanism", dce`
	`sp_serveroption rem_server, "use message integrity", true`
		"Setting server connection options" on page 505.
		sp_serveroption in the *Reference Manual*.

Example of setting up security model B for RPCs

Assume that:

- A local server, lcl_serv, will run RPCs on a remote server, rem_serv.
- Both servers will use security model B and the DCE security service.
- These RPC security services will be in effect: mutual authentication and message integrity.
- Users "user1" and "user2" will use unified login to log in to the local server, lcl_serv, and run RPCs on rem_serv. These users will be "trusted" on rem_serv and will not need to specify a password for the remote server.
- User "user3" will not use unified login, will not be trusted, and must supply a password to Adaptive Server when logging in.

You would use the following sequence of commands to set up security for RPCs between the servers:

System Security Officer on remote server (rem_serv):

```
sp_addserver 'lcl_serv'
sp_addlogin user1, "eracg12"
sp_addlogin user2, "esirpret"
sp_addlogin user3, "drabmok"
sp_configure "use security services", 1
sp_serveroption lcl_serv, "rpc security model B",
   true
sp_serveroption lcl_serv, "security mechanism", dce
```

System Administrator on remote server (rem_serv):

```
sp_remoteoption lcl_serv, user1, user1, trusted,
   true
sp_remoteoption lcl_serv, user2, user2, trusted,
   true
```

System Security Officer on local server (lcl_serv):

```
sp_addserver lcl_serv, local
sp_addserver rem_serv
sp_addlogin user1, "eracg12"
sp_addlogin user2, "esirpret"
sp_addlogin user3, "drabmo1"
sp_configure "use security services", 1
sp_configure rem_serv, "rpc security model B", true
sp_serveroption rem_serv, "security mechanism", dce
sp_serveroption rem_serv, "mutual authentication"
   true
sp_serveroption rem_serv, "use message integrity"
   true
```

In addition, the *interfaces* file or Directory Service must have entries for rem_serv and lcl_serv. Each entry should specify the "dce" security service. For example, you might have these *interfaces* entries, as created by the dscp utility:

```
## lcl_serv (3201)
lcl_serv
master tli tcp /dev/tcp \x00020c8182d655110000000000000000
query tli tcp /dev/tcp \x00020c8182d65511000000000000000
secmech 1.3.6.1.4.1.897.4.6.1
## rem_serv (3519)
rem_serv
master tli tcp /dev/tcp \x000214ad82d655110000000000000000
query tli tcp /dev/tcp \x000214ad82d65511000000000000000
secmech 1.3.6.1.4.1.897.4.6.1
```

Note To actually use the security services on either server, you must restart the server so that the static parameter, use security services, takes effect.

For detailed information about setting up servers for remote procedure calls, see Chapter 13, "Managing Remote Servers."

Getting information about remote servers

The system procedure sp_helpserver displays information about servers. When it is used 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:

- -K *keytab_file*
- -R *remote_server_principal*
- -V *security_options*
- -Z *security_mechanism*

Note Versions of isql and bcp for the DCE Directory Service and for DCE security services are available. They are isql_dce and bcp_dce. You must use these versions when you are using DCE.

These options are described in the following paragraphs.

-K *keytab_file* – can be used only with DCE security. It specifies a DCE keytab file that contains the security key for the user logging into the server. Keytab files can be created with the DCE dcecp utility—see your DCE documentation for more information.

If the -K option is not supplied, the user of isql must be logged into DCE. If the user specifies the -U option, the name specified with -U must match the name defined for the user in DCE.

-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 into 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 – enable data confidentiality service
- i – enable data integrity service
- m – enable mutual authentication for connection establishment
- o – enable data origin stamping service
- r – enable data replay detection
- q – enable out-of-sequence detection

-Z *security_mechanism* – specifies the name of a security mechanism to use on the connection.

Security mechanism names are defined in the *libtcl.cfg* configuration file. If no *security_mechanism* name is supplied, the default mechanism is used. For more information about security mechanism names, see the *Open Client/Server Configuration Guide* for your platform.

If you log in to the security mechanism and then log in to Adaptive Server, you do not need to specify the -U option on the utility because Adaptive Server gets the user name from the security mechanism. For example, consider the following session:

```
svrsole4% dce_login user2
Enter Password:
svrsole4% $SYBASE/bin/isql_dce -V
1> select suser_name()
2> go

------------------------------
user2
```

For this example, "user2" logs in to DCE with dce_login and then logs into Adaptive Server without specifying the -U option. The -V option without parameters implicitly specifies one security service: unified login.

For more information about Adaptive Server utilities, see the *Utility Guide*.

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 CS_SEC_DETECTSEQ property. For information about using security services with Client-Library, see the *Open Client Client-Library/C Reference Manual*.

Example of using security services

Assume that your login is "mary" and you want to use the DCE security mechanism with unified login (always in effect when you specify the -V option of isql_dce or bcp_dce), message confidentiality, and mutual authentication for remote procedures. You want to connect to server WOND and run remote procedures on GATEWAY with mutual authentication. Assuming that a System Security Officer has set up both WOND and GATEWAY for rpc Model B, added you as a user on both servers, and defined you as a remote, "trusted" user on GATEWAY, you can use the following process:

1. Log in to the DCE security mechanism and receive a credential:

   ```
   dce_login mary
   ```

2. Log in to the Adaptive Server with isql_dce:

   ```
   isql_dce -SWOND -Vcm
   ```

3. Run:

   ```
   GATEWAY...sp_who
   GATEWAY...mary_prc1
   GATEWAY...mary_prc2
   ```

Now, all messages that Mary sends to the server and receives from the server will be encrypted (message confidentiality), and when she runs remote procedures, both the WOND and GATEWAY servers will be authenticated.

Using security mechanisms for the client

Adaptive Server, when it is started, determines the set of security mechanisms it supports. (See "Determining security mechanisms to support" on page 533. From the list of security mechanisms that Adaptive Server supports, it must choose the one to be used for a particular client.

If the client specifies a security mechanism (for example with the -Z option of isql_dce), Adaptive Server uses that security mechanism. Otherwise, it uses the first security mechanism listed in the *libtcl.cfg* file.

Getting information about available security services

Adaptive Server enables you to:

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

Determining supported security services and mechanisms

A system table, syssecmechs, provides information about the security mechanisms and security services supported 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 "dce" or "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."

Several rows may be in the table 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 this query:

```
select * from syssecmechs
```

The result might look something like:

```
sec_mech_name                   available_service
------------------------------ --------------------
 dce                            unifiedlogin
 dce                            mutualauth
 dce                            delegation
 dce                            integrity
 dce                            confidentiality
 dce                            detectreplay
 dce                            detectseq
```

Determining enabled security services

To determine which security services are enabled for the current session, use the function show_sec_services. For example:

```
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. The following is the syntax, where *security_service_nm* is a security service that is available:

is_sec_service_on(*security_service_nm*)

Use the name that is displayed when you query syssecmechs.

For example, to determine whether “mutualauth” is enabled, execute:

```
select is_sec_service_on("mutualauth")
```

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

A result of 1 indicates the security service is enabled for the session. A result of 0 indicates the service is not in use.

CHAPTER 15 Overview of Disk Resource Issues

This chapter discusses some basic issues that determine how you allocate and use disk resources with Adaptive Server.

Many Adaptive Server defaults are set to reasonable values for aspects of storage management, such as where databases, tables, and indexes are placed and how much space is allocated for each one. Responsibility for storage allocation and management is often centralized, and usually, the System Administrator has ultimate control over the allocation of disk resources to Adaptive Server and the physical placement of databases, tables, and indexes on those resources.

Device allocation and object placement

When configuring a new system, the System Administrator must consider several issues that have a direct impact on the number and size of disk resources required. These device allocation issues refer to commands and procedures that add disk resources to Adaptive Server. Device allocation topics are described in the chapters shown in Table 15-1.

Table 15-1: Device allocation topics

Task	Chapter
Initialize and allocate a default pool of database devices.	Chapter 16, “Initializing Database Devices”
Mirror database devices for recovery.	Chapter 17, “Mirroring Database Devices”

After the initial disk resources have been allocated to Adaptive Server, the System Administrator, Database Owner, and object owners should consider how to place databases and database objects on specific database devices. These object placement issues determine where database objects reside on your system and whether or not the objects share devices. Object placement tasks are discussed throughout this manual, including the chapters shown in Table 15-2.

Table 15-2: Object placement topics

Task	Chapter
Place databases on specific database devices.	Chapter 21, "Creating and Managing User Databases"
Place tables and indexes on specific database devices.	Chapter 24, "Creating and Using Segments"

Do not consider allocating devices separately from object placement. For example, if you decide that a particular table must reside on a dedicated pair of devices, you must first allocate those devices to Adaptive Server. The remaining sections in this chapter provide an overview that spans both device allocation and object placement issues, providing pointers to chapters where appropriate.

Commands for managing disk resources

Table 15-3 lists the major commands a System Administrator uses to allocate disk resources to Adaptive Server and provides references to the chapters that discuss those commands.

Table 15-3: Commands for allocating disk resources

Command	Task	See

disk init
name = "*dev_name*"
physname = "*phys_name*"...
	Makes a physical device available to a particular Adaptive Server. Assigns a database device name (*dev_name*) that is used to identify the device in other Adaptive Server commands.	Chapter 16, "Initializing Database Devices"

sp_deviceattr *logicalname*, *optname*, *optvalue*	Changes the *dsync* setting of an existing database device file	Chapter 16, "Initializing Database Devices"
sp_diskdefault "*dev_name*"...	Adds *dev_name* to the general pool of default database space.	Chapter 16, "Initializing Database Devices"

disk resize
name = "*device_name*",
size = *additional_space*
	Dynamically increases the size of database devices.	Chapter 16, "Initializing Database Devices"

Command	Task	See

disk mirror
name = "*dev_name*"
mirror = "*phys_name*"...
	Mirrors a database device on a specific physical device.	Chapter 17, “Mirroring Database Devices”

Table 15-4 lists the commands used in object placement. For information about how object placement affects performance, see Chapter 6, “Controlling Physical Data Placement,” in the *Performance and Tuning: Basics*.

Table 15-4: Commands for placing objects on disk resources

Command	Task	See
create database...on *dev_name* or alter database...on *dev_name*	Makes database devices available to a particular Adaptive Server database. The log on clause to create database places the database’s logs on a particular database device.	Chapter 21, “Creating and Managing User Databases”
create database... or alter database...	When used without the on *dev_name* clause, these commands allocate space on the default database devices.	Chapter 21, “Creating and Managing User Databases”
sp_addsegment *seg_name*, *dbname*, *devname* and sp_extendsegment *seg_name*, *dbname*, *devname*	Creates a segment, a named collection of space, from the devices available to a particular database.	Chapter 24, “Creating and Using Segments”
create table...on *seg_name* or create index...on *seg_name*	Creates database objects, placing them on a specific segment of the database’s assigned disk space.	Chapter 24, “Creating and Using Segments”
create table... or create index...	When used without on *seg_name*, tables and indexes occupy the general pool of space allocated to the database (the default devices).	Chapter 24, “Creating and Using Segments”

Considerations in storage management decisions

The System Administrator must make many decisions regarding the physical allocation of space to Adaptive Server databases. The major considerations in these choices are:

- **Recovery** – disk mirroring and maintaining logs on a separate physical device provide two mechanisms for full recovery in the event of physical disk crashes.

- **Performance** – for tables or databases where speed of disk reads and writes is crucial, properly placing database objects on physical devices yields performance improvements. Disk mirroring slows the speed of disk writes.

Recovery

Recovery is the key motivation for using several disk devices. Nonstop recovery can be accomplished by mirroring database devices. Full recovery can also be ensured by storing a database's log on a separate physical device.

Keeping logs on a separate device

Unless a database device is mirrored, full recovery requires that a database's transaction log be stored on a different device from the actual data (including indexes) of a database. In the event of a hard disk crash, you can create an up-to-date database by loading a dump of the database and then applying the log records that were safely stored on another device. See Chapter 21, "Creating and Managing User Databases," for information about the log on clause of create database.

Mirroring

Nonstop recovery in the event of a hard disk crash is guaranteed by of mirroring all Adaptive Server devices to a separate physical disk. Chapter 17, "Mirroring Database Devices," describes the process of mirroring devices.

Performance

You can improve system performance by placing logs and database objects on separate devices:

- Placing a table on one hard disk and nonclustered indexes on another ensures that physical reads and writes are faster, since the work is split between two disk drives.
- Splitting large tables across two disks can improve performance, particularly for multi-user applications.
- When log and data share devices, user log cache buffering of transaction log records is disabled.

- Partitioning provides multiple insertion points for a heap table, adds a degree of parallelism to systems configured to perform parallel query processing, and makes it possible to distribute a table's I/O across multiple database devices.

See Chapter 6, "Controlling Physical Data Placement," in *Performance and Tuning: Basics* for a detailed discussion of how object placement affects performance.

Status and defaults at installation time

You can find instructions for installing Adaptive Server in the installation documentation for your platform. The installation program and scripts initialize the master device and set up the master, model, sybsystemprocs, sybsecurity, and temporary databases for you.

When you install Adaptive Server, the system databases, system defined segments, and database devices are organized as follows:

- The master, model, and tempdb databases are installed on the master device.
- The sybsystemprocs database is installed on a device that you specified.
- Three segments are created in each database: system, default, and logsegment.
- The master device is the default storage device for all user-created databases.

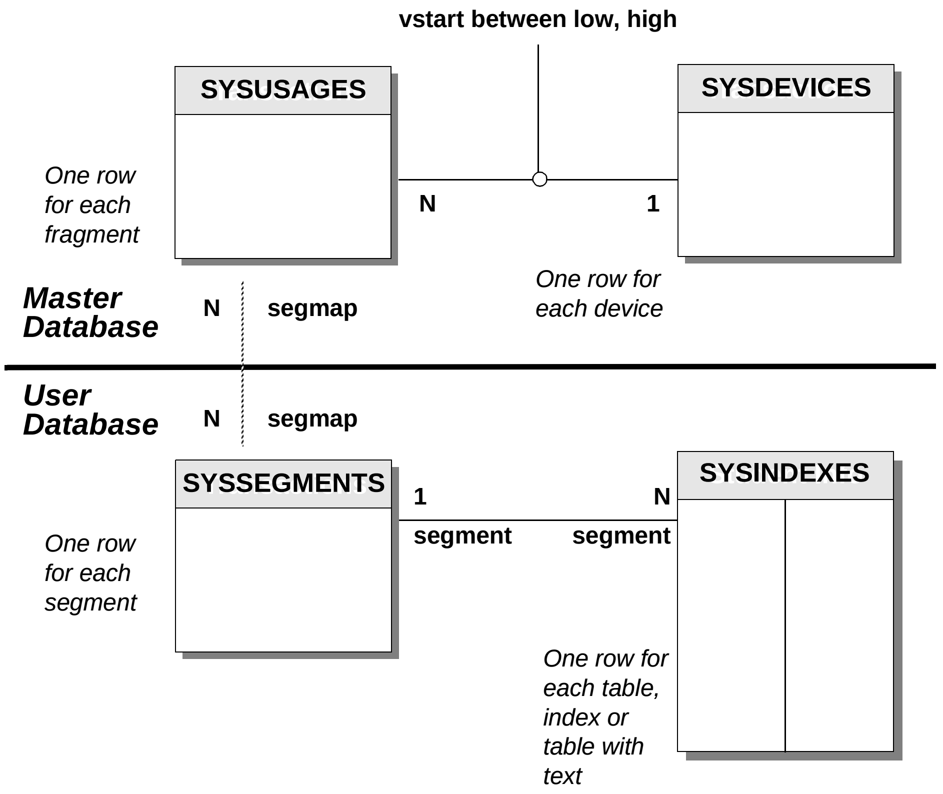
Note After initializing new devices for default storage, remove the master device from the default storage area with sp_diskdefault. Do not store user databases and objects on the master device. See "Designating default devices" on page 566 for more information.

- If you install the audit database, sybsecurity, it is located on its own device.

System tables that manage storage

Two system tables in the master database and two more in each user database track the placement of databases, tables (including the transaction log table, syslogs), and indexes. The relationship between the tables is illustrated in Figure 15-1.

Figure 15-1: System tables that manage storage



The *sysdevices* table

The sysdevices table in the master database contains one row for each **database device** and may contain a row for each dump device (tape, disk, or operating system file) available to Adaptive Server.

The disk init command adds entries for database devices to master..sysdevices. Dump devices, added with the system procedure sp_addumpdevice, are discussed in Chapter 27, “Developing a Backup and Recovery Plan.”

sysdevices stores two names for each device:

- A *logical name* or *device name*, used in all subsequent storage-management commands, is stored in the name column of sysdevices. This is usually a user-friendly name, perhaps indicating the planned use for the device, for example “logdev” or “userdbdev.”
- The *physical name* is the actual operating system name of the device. You use this name only in the disk init command; after that, all Adaptive Server data storage commands use the logical name.

You place a database or transaction log on one or more devices by specifying the logical name of the device in the create database or alter database statement. The log on clause to create database places a database’s transaction log on a separate device to ensure full recoverability. The log device must also have an entry in sysdevices before you can use log on.

A database can reside on one or more devices, and a device can store one or more databases. See Chapter 21, “Creating and Managing User Databases,” for information about creating databases on specific database devices.

The *sysusages* table

The sysusages table in the master database keeps track of all of the space that you assign to all Adaptive Server databases.

create database and alter database allocate new space to the database by adding a row to sysusages for each database device or device fragment. When you allocate only a portion of the space on a device with create or alter database, that portion is called a *fragment*.

The system procedures sp_addsegment, sp_dropsegment, and sp_extendsegment change the segmap column in sysusages for the device that is mapped or unmapped to a segment. Chapter 24, “Creating and Using Segments,” discusses these procedures in detail.

The *syssegments* table

The syssegments table, one in each database, lists the segments in a database. A **segment** is a collection of the database devices and/or fragments available to a particular database. Tables and indexes can be assigned to a particular segment, and therefore to a particular physical device, or can span a set of physical devices.

create database makes default entries in syssegments. The system procedures sp_addsegment and sp_dropsegment add and remove entries from syssegments.

The *sysindexes* table

The sysindexes table lists each table and index and the segment where each table, clustered index, nonclustered index, and chain of text pages is stored. It also lists other information such as the max_rows_per_page setting for the table or index.

The create table, create index, and alter table commands create new rows in sysindexes. Partitioning a table changes the function of sysindexes entries for the table, as described in *Performance and Tuning: Basics*.

CHAPTER 16 Initializing Database Devices

This chapter explains how to initialize database devices and how to assign devices to the default pool of devices.

What are database devices?

A database device stores the objects that make up databases. The term *device* does not necessarily refer to a distinct physical device: it can refer to any piece of a disk (such as a disk partition) or a file in the file system that is used to store databases and their objects.

Each database device or file must be prepared and made known to Adaptive Server before it can be used for database storage. This process is called **initialization**.

After a database device has been initialized, it can be:

- Allocated to the default pool of devices for the create and alter database commands
- Assigned to the pool of space available to a user database
- Assigned to a user database and used to store one or more database objects
- Assigned to store a database's transaction logs

Using the *disk init* command

A System Administrator initializes new database devices with the disk init command, which:

- Maps the specified physical disk device or operating system file to a *database device* name
- Lists the new device in master..sysdevices
- Prepares the device for database storage

Note Before you run disk init, see the installation documentation for your platform for information about choosing a database device and preparing it for use with Adaptive Server. You may want to repartition the disks on your computer to provide maximum performance for your Sybase databases.

disk init divides the database devices into **allocation units**. The size of the allocation unit depends on which logical page size your server is configured for (2, 4, 8, or 16K). In each allocation unit, the disk init command initializes the first page as the allocation page, which will contain information about the database (if any) that resides on the allocation unit.

Warning! After you run the disk init command, dump the master database. This makes recovery easier and safer in case master is damaged. See Chapter 29, “Restoring the System Databases.”

disk init syntax

The syntax of disk init is:

```
disk init
    name = "device_name" ,
    physname = "physicalname" ,
    [vdevno = virtual_device_number , ]
    size = size_of_device
    [, vstart = virtual_address ,
        cntrltype = controller_number]
    [, dsync = {true | false}]
```

disk init **examples**

On UNIX:

```
disk init
  name = "user_disk",
  physname = "/dev/rxy1a",
  size = "10M"
```

On Windows NT:

```
disk init
  name = "user_disk",
  physname = "d:\devices\userdisk.dat",
  size = "10M"
```

Specifying a logical device name with *disk init*

The *device_name* must be a valid identifier. This name is used in the create database and alter database commands, and in the system procedures that manage segments. The logical device name is known only to Adaptive Server, not to the operating system on which the server runs.

Specifying a physical device name with *disk init*

The *physicalname* of the database device gives the name of a raw disk partition (UNIX), foreign device, or the name of an operating system file. On PC platforms, you typically use operating system file names for *physicalname*.

Choosing a device number for *disk init*

Adaptive Server automatically specifies the next available identifying number for the database device. This is the virtual device number (vdevno). You do not have to specify this number when you issue the disk init command.

If you choose to manually select the vdevno, it must be unique among the devices used by Adaptive Server. Device number 0 represents the master device. The highest number must be one less than the number of database devices for which your system is configured. For example, for a system with a default configuration of 10 devices, the legal device numbers are 1–9. To see the configuration value for your system, execute sp_configure "number of devices" and check the run value:

```
                       sp_configure "number of devices"
Parameter name  Default  Memory Used  Config Value  Run Value
--------------- -------  -----------  ------------  ----------
number of devices    10            0            10         10
```

To see the numbers already in use for vdevno, look in the device_number column of the report from sp_helpdevice, or use the following query to list all the device numbers currently in use:

```
select distinct low/16777216
    from sysdevices
    order by low
```

Adaptive Server is originally configured for 10 devices. You may be limited to a smaller number of devices by operating system constraints. See the discussion of sp_configure, which is used to change configuration parameters, in Chapter 4, "Setting Configuration Parameters."

Specifying the device size with *disk init*

You can use the following unit specifiers to indicate the size of the device: 'k' or 'K' indicate kilobytes, 'm' or 'M' indicate megabytes and 'g' or 'G' indicate gigabytes. Although it is optional, Sybase recommends that you always include the unit specifier in both the disk init and create database commands to avoid confusion in the actual number of pages allocated. You must enclose the unit specifier in single or double quotes.

The following apply to the syntax for disk init:

- You can specify the size as a float, but it is rounded down to the nearest whole value. For example, if you specify a size of 3.75G, it is rounded down to 3G.
- If you do not specify a size:
    - disk init and disk reinit use the basic disk page size of 2K.
    - The size argument for create database and alter database is in terms of megabytes of disk piece. This value is converted to the number of logical page size that with which the master device was built.
    - Minimum size of a database. You cannot alter the size of a database device after running disk init.

- If you are planning to use the new device for the creation of a new database, the minimum size depends on the logical page size used by the server, described in Table 16-1:

Table 16-1: Minimum database sizes

Logical page size	Minimum database size
2K	2 Megabytes
4K	4 Megabytes
8K	8 Megabytes
16K	16 Megabytes

If you are initializing a database device for a transaction log or for storing small tables or indexes on a segment, the size can be as small as 512 blocks (1MB).

If you are initializing a raw device, determine the size of the device from your operating system, as described in the the installation documentation for your platform. Use the total size available, up to the maximum for your platform. After you have initialized the disk for use by Adaptive Server, you cannot use any space on the disk for any other purpose.

disk init uses size to compute the value for the high virtual page number in sysdevices.high.

Warning! If the physical device does not contain the number of blocks specified by the size parameter, the disk init command fails. If you use the optional vstart parameter, the physical device must contain the sum of the blocks specified by both the vstart and size parameters, or the command fails.

Specifying the *dsync* setting with *disk init* (optional)

For devices initialized on UNIX operating system files, the dsync setting controls whether or not writes to those files are buffered. When the dsync setting is on, Adaptive Server opens a database device file using the UNIX dsync flag. The dsync flag ensures that writes to the device file occur directly on the physical storage media, and Adaptive Server can recover data on the device in the event of a system failure.

When dsync is off, writes to the device file may be buffered by the UNIX and Windows NT file systems, and the recovery of data on the device cannot be ensured. The dsync setting should be turned off only when data integrity is not required, or when the System Administrator requires performance and behavior similar to earlier Adaptive Server versions.

Note The dsync setting is ignored for devices initialized on raw partitions, and for devices initialized on Windows NT files. In both cases, writes to the database device take place directly to the physical media.

Performance implications of *dsync*

The use of the dsync setting with database device files incurs the following performance trade-offs:

- Adaptive Server does not support asynchronous I/O on operating system files for HP-UX and Digital UNIX.
- If database device files on these platforms use the dsync option, then the Adaptive Server engine writing to the device file will block until the write operation completes. This can cause poor performance during update operations.
- When dsync is on, write operations to database device files may be slower compared to previous versions of Adaptive Server (where dsync is not supported). This is because Adaptive Server must write data to disk instead of simply copying cached data to the UNIX file system buffer.

  In cases where highest write performance is required (but data integrity after a system failure is not required) turning dsync off yields device file performance similar to earlier Adaptive Server versions. For example, you may consider storing tempdb on a dedicated device file with dsync disabled, if performance is not acceptable while using dsync.
- Response time for read operations is generally better for devices stored on UNIX operating system files as compared to devices stored on raw partitions. Data from device files can benefit from the UNIX file system cache as well as the Adaptive Server cache, and more reads may take place without requiring physical disk access.
- The disk init command takes longer to complete with previous Adaptive Server versions, because the required disk space is allocated during device initialization.

Limitations and restrictions of *dsync*

The following limitations and restrictions apply to using the dsync setting:

- dsync is always set to “true” for the master device file. You cannot change the dsync setting for the master device. If you attempt to turn dsync off for the master device, Adaptive Server displays a warning message.
- If you change a device file’s dsync setting using the sp_deviceattr procedure, you must restart Adaptive Server before the change takes affect.
- When you upgrade from an Adaptive Server prior to version 12.x, dsync is set to “true” for the master device file only. You must use the sp_deviceattr procedure to change the dsync setting for any other device files.
- Adaptive Server ignores the dsync setting for database devices stored on raw partitions. Writes to devices stored on raw partitions are always done directly to the physical media.
- Adaptive Server also ignores the dsync setting for database devices stored on Windows NT operating system files. Adaptive Server on Windows NT automatically uses a capability similar to dsync for all database device files.

Other optional parameters for *disk init*

vstart is the starting virtual address, or the offset, for Adaptive Server to begin using the database device. vstart accepts the following optional unit specifiers: k or K (kilobytes), m or M (megabytes), and g or G (gigabytes). The size of the offset depends on how you enter the value for vstart.

- If you do not specify a unit size, vstart uses 2K pages for its starting address. For example, if you specify `vstart = 13`, Adaptive Server uses 13 * 2K pages as the offset for the starting address.
- If you specify a unit value, vstart uses this as the starting address. For example, if you specify `vstart = "13M"`, Adaptive Server sets the starting address offset at 13 megabytes.

The default value (and usually the preferred value) of vstart is 0. If the specified device does not have the sum of vstart + size blocks available, the disk init command fails.

The optional cntrltype keyword specifies the disk controller. Its default value is 0. Reset it only if instructed to do so.

Note To perform disk initialization, the user who started Adaptive Server must have the appropriate operating system permissions on the device that is being initialized.

Getting information about devices

The system procedure sp_helpdevice provides information about the devices in the sysdevices table.

When used without a device name, sp_helpdevice lists all the devices available on Adaptive Server. When used with a device name, it lists information about that device. Here, sp_helpdevice is used to report information about the master device:

```
                             sp_helpdevice master
device_name  physical_name  description
-----------  -------------- ----------------------------------------
master        d_master         special, default disk, physical disk, 20 MB


status      cntrltype    device_number     low      high
------      ----------   -------------     ------   -------
3           0            0                 0        9999
```

Each row in master..sysdevices describes:

- A dump device (tape, disk, or file) to be used for backing up databases, or
- A database device to be used for database storage.

The initial contents of sysdevices are operating-system-dependent. Entries in sysdevices usually include:

- One for the master device
- One for the sybsystemprocs database, which you can use to store additional databases such as pubs2 and sybsyntax, or for user databases and logs
- Two for tape dump devices

If you installed auditing, there will also be a separate device for sybsecurity.

The low and high fields represent the page numbers that have been assigned to the device. For dump devices, they represent the media capacity of the device.

The status field in sysdevices is a bitmap that indicates the type of device, whether a disk device will be used as a default storage device when users issue a create or alter database command without specifying a database device, disk mirroring information, and dsync settings. The status bits and their meanings are listed in Table 16-2:

Table 16-2: Status bits in sysdevices

Bit	Meaning
1	Default disk (may be used by any create or alter database command that does not specify a location)
2	Physical disk
4	Logical disk (not used)
8	Skip header (used with tape dump devices)
16	Dump device
32	Serial writes
64	Device mirrored
128	Reads mirrored
256	Secondary mirror side only
512	Mirror enabled
2048	Used internally; set after disk unmirror, side = retain
4096	Primary device needs to be unmirrored (used internally)
8192	Secondary device needs to be unmirrored (used internally)
16384	UNIX file device uses dsync setting (writes occur directly to physical media)

For more information about dump devices and sp_addumpdevice, see Chapter 27, “Developing a Backup and Recovery Plan.”

Dropping devices

To drop database and dump devices, use sp_dropdevice. The syntax is:

sp_dropdevice *logicalname*

You cannot drop a device that is in use by a database. You must drop the database first.

sp_dropdevice removes the device name from sysdevices. sp_dropdevice does not remove an operating system file: it only makes the file inaccessible to Adaptive Server. You must use operating system commands to delete a file after using sp_dropdevice.

Designating default devices

To create a pool of default database devices to be used by all Adaptive Server users for creating databases, use sp_diskdefault after the devices are initialized. sp_diskdefault marks these devices in sysdevices as default devices. Whenever users create (or alter) databases without specifying a database device, new disk space is allocated from the pool of default disk space.

The syntax for sp_diskdefault is:

sp_diskdefault *logicalname*, {defaulton | defaultoff}

You are most likely to use the defaultoff option to remove the master device from the pool of default space:

```
sp_diskdefault master, defaultoff
```

The following command makes sprocdev, the device that holds the sybsystemprocs database, a default device:

```
sp_diskdefault sprocdev, defaulton
```

Adaptive Server can have multiple default devices. They are used in the order in which they appear in the sysdevices table (that is, alphabetical order). When the first default device is filled, the second default device is used, and so on.

Note After initializing a set of database devices, you may want to assign them to specific databases or database objects rather than adding them to the default pool of devices. For example, you may want to make sure a table never grows beyond the size of a particular device.

Choosing default and nondefault devices

sp_diskdefault lets you plan space usage carefully for performance and recovery, while allowing users to create or alter databases.

Make sure these devices are *not* default devices:

- The master device (use sp_diskdefault to set defaultoff after adding user devices)
- The device for sybsecurity
- Any device intended solely for logs
- Devices where high-performance databases reside, perhaps using segments

You can use the device that holds sybsystemprocs for other user databases.

Note If you are using disk mirroring or segments, you should exercise caution in deciding which devices you add to the default list with sp_diskdefault. In most cases, devices that are to be mirrored or databases that will contain objects placed on segments should allocate devices specifically, rather than being made part of default storage.

Increasing the size of devices with *disk resize*

The disk resize command allows you to dynamically increase the size of your database devices, rather than initializing a new device. For example, if /sybase/testdev.dat requires an additional 10MB of space, you can run disk resize and allocate this amount of space to the device. The create and alter database commands can use this added space.

You can use disk resize to increase the size for both devices on raw partitions and for file systems. The minimum amount of space that you can increase a device is 1MB or an allocation unit, whichever is greater.

You cannot use disk resize on dump or load devices.

Any properties that are set on the device continue to be set after you increase its size. That is, if a device has dsync set before you increase its size, it will have dsync set afterwards. Also, any access rights that were set before you increased the size of the device remain set.

Only users with the sa role can execute the disk resize command.

You can use audit trails on disk resize to track the number of times a device is resized. The device being resized is always online and available for users during the resize operation.

See Chapter 7, “Commands” in the *Reference Manual* for syntax information about disk resize.

Using the *disk resize* command

A user with the sa role can execute the disk resize command, which:

- Updates the high value in master....sysdevices, and
- Prepares the additional space for database storage.

Insufficient disk space

During the physical initialization of the disk, if an error occurs due to insufficient disk space, disk resize extends the database device to the largest size possible before the error occurs.

For example, on a server that uses 4K logical pages, if you try to increase the size of the device by 40MB, but only 39.5MB is available, the device is extended only by 39.5MB.

Device shrinkage

You cannot decrease the size of a device with disk resize.

disk resize syntax

disk resize has the following syntax:

```
disk resize
   name = “device_name”,
   size = additional_space
```

Where *device_name* is the name of the device you are increasing and *additional_space* is the additional disk space you are adding to this device.

- You must have already initialized the device with disk init.
- *device_name* must refer to a valid logical device name.

- The minimum size for disk resize is 1MB or one allocation unit, whichever is greater.
- You must permanently disable mirroring while the resize operation is in progress. You can reestablish mirroring when the resize operation is complete.

Page size	Allocation unit size	Minimum incremental size
2K	0.5MB	1MB
4K	1MB	1MB
8K	2MB	2MB
16K	4MB	4MB
32K	8MB	8MB
64K	16MB	16MB

Note The new size of the device is the sum of the old device size plus the size specified in the disk resize command.

Disk resize example

For example, the configuration of the device testdev from isql:

```
sp_helpdevice testdev
device_name  physical_name          description
  status   cntrltype  device_number   low              high
-----------  -------------------    -------------
  -------  ---------  ------------    -------------    -------------
testdev      /sybase/dev/testdev.dat   special, dsync on, physical disk, 4.00MB
  16386    0          8                134217728.000000    134219775.000000
```

To increase the size of testdev by 4MB using disk resize, enter:

```
disk resize
name = "test_dev",
size = "4M"
```

testdev.dat is now 8MB:

```
sp_helpdevice testdev
device_name  physical_name          description
  status   cntrltype  device_number   low              high
-----------  -------------------    -------------
```

```
  ------- ---------  ------------   ------------     -------------
testdev      /sybase/dev/testdev.dat   special, dsync on, physical disk, 8.00MB
  16386    0           8               134217728.000000    134221823.000000
```

Specifying a logical device name with *disk resize*

The *device_name* must have a valid identifier. The device should have already been initialized using the disk init command and it must refer to a valid Adaptive Server device.

Specifying the device size with *disk resize*

You can use the following unit specifiers to indicate the size of the device: “k” or “K” to indicate kilobytes, “m” or “M” to indicate megabytes and “g” or “G” to indicate gigabytes.

Although it is optional, Sybase recommends that you always include the unit specifier with the disk resize command to avoid confusion in the actual number of pages allocated. You must enclose the unit specifier in single or double quotes. If you do not use a unit specifier, the size defaults to the number of disk pages.

To verify the new size, use sp_helpdevice.

CHAPTER 17 Mirroring Database Devices

This chapter describes creating and administering disk mirrors.

What is disk mirroring?

Disk mirroring can provide nonstop recovery in the event of media failure. The disk mirror command causes an Adaptive Server database device to be duplicated, that is, all writes to the device are copied to a separate physical device. If one device fails, the other contains an up-to-date copy of all transactions.

When a read or write to a mirrored device fails, Adaptive Server "unmirrors" the bad device and displays error messages. Adaptive Server continues to run unmirrored.

Deciding what to mirror

When deciding to mirror a device, you must weigh such factors as the costs of system downtime, possible reduction in performance, and the cost of storage media. Reviewing these issues will help you decide what to mirror—just the transaction logs, all devices on a server, or selected devices.

Note You cannot mirror a dump device.

You should mirror all default database devices so that you are protected if a create or alter database command affects a database device in the default list.

In addition to mirroring user database devices, you should always put their transaction logs on a separate database device. You can also mirror the database device used for transaction logs for even greater protection.

To put a database's transaction log (that is, the system table syslogs) on a different device than the one on which the rest of the database is stored, name the database device and the log device when you create the database. You can also use alter database to add a second device and then run the system procedure sp_logdevice.

Here are three examples that involve different cost and performance trade-offs:

- *Speed of recovery* – you can achieve nonstop recovery when the master and user databases (including logs) are mirrored and can recover without the need to reload transaction logs.
- *Storage space* – immediate recovery requires full redundancy (all databases and logs mirrored), which consumes disk space.
- *Impact on performance* – Mirroring the user databases (as shown in Figure 17-2 and Figure 17-3) increases the time needed to write transactions to both disks.

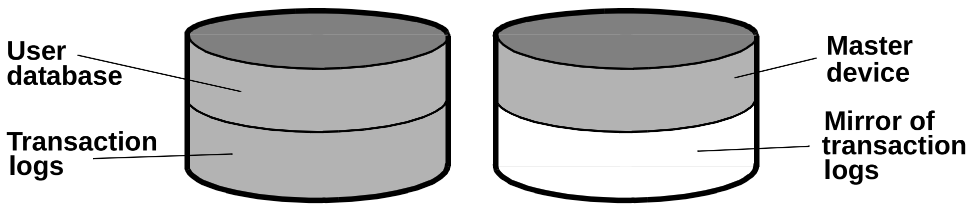
Mirroring using minimal physical disk space

Figure 17-1 illustrates the "minimum guaranteed configuration" for database recovery in case of hardware failure. The master device and a mirror of the user database transaction log are stored in separate partitions on one physical disk. The other disk stores the user database and its transaction log in two separate disk partitions.

If the disk with the user database fails, you can restore the user database on a new disk from your backups and the mirrored transaction log.

If the disk with the master device fails, you can restore the master device from a database dump of the master database and remirror the user database's transaction log.

Figure 17-1: Disk mirroring using minimal physical disk space



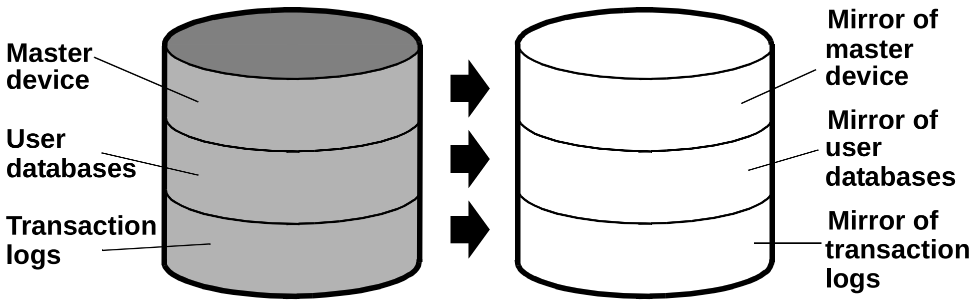
This configuration minimizes the amount of disk storage required. It provides for full recovery, even if the disk storing the user database and transaction log is damaged, because the mirror of the transaction log ensures full recovery. However, this configuration does not provide nonstop recovery because the master and user databases are not being mirrored and must be recovered from backups.

Mirroring for nonstop recovery

Figure 17-2 represents another mirror configuration. In this case, the master device, user databases, and transaction log are all stored on different partitions of the same physical device and are all mirrored to a second physical device.

The configuration in Figure 17-2 provides nonstop recovery from hardware failure. Working copies of the master and user databases and log on the primary disk are all being mirrored, and failure of either disk will not interrupt Adaptive Server users.

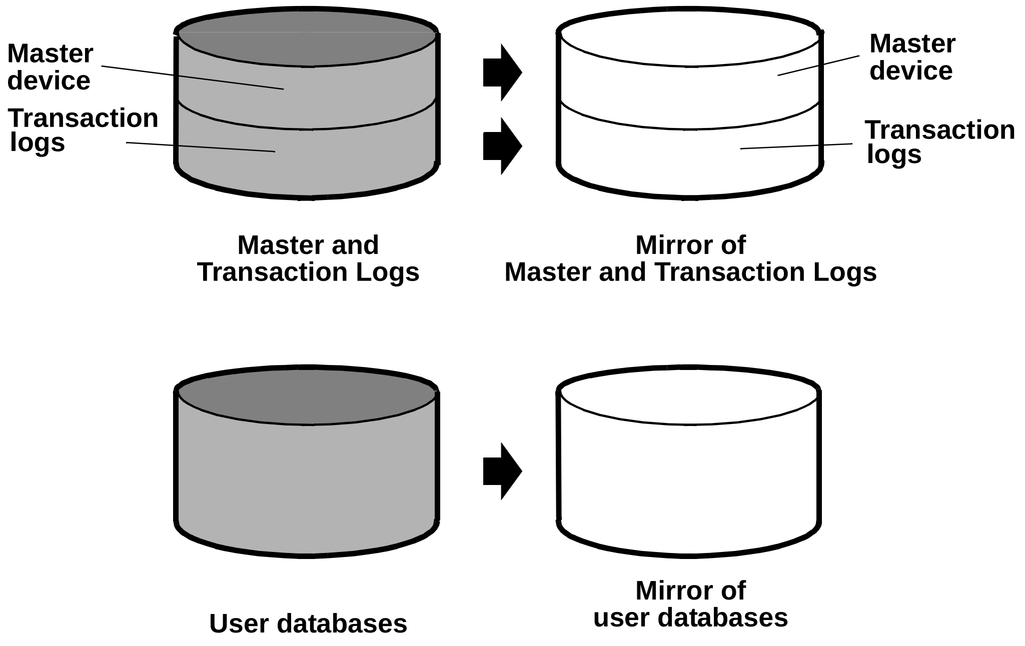
Figure 17-2: Disk mirroring for rapid recovery



With this configuration, all data is written twice, once to the primary disk and once to the mirror. Applications that involve many writes may be slower with disk mirroring than without mirroring.

Figure 17-3 illustrates another configuration with a high level of redundancy. In this configuration, all three database devices are mirrored, but the configuration uses four disks instead of two. This configuration speeds performance during write transactions because the database transaction log is stored on a different device from the user databases, and the system can access both with less disk head travel.

Figure 17-3: Disk mirroring: keeping transaction logs on a separate disk



Conditions that do not disable mirroring

Adaptive Server disables a mirror only when it encounters an I/O error on a mirrored device. For example, if Adaptive Server tries to write to a bad block on the disk, the resulting error disables mirroring for the device. However, processing continues without interruption on the unaffected mirror.

The following conditions *do not* disable a mirror:

- An unused block on a device is bad. Adaptive Server does not detect an I/O error and disables mirroring until it accesses the bad block.
- Data on a device is overwritten. This might happen if a mirrored device is mounted as a UNIX file system, and UNIX overwrites the Adaptive Server data. This causes database corruption, but mirroring is not disabled, since Adaptive Server would not encounter an I/O error.
- Incorrect data is written to both the primary and secondary devices.
- The file permissions on an active device are changed. Some System Administrators may try to test disk mirroring by changing permissions on one device, hoping to trigger I/O failure and unmirror the other device. But the UNIX operating system does not check permissions on a device after opening it, so the I/O failure does not occur until the next time the device is started.

Disk mirroring is not designed to detect or prevent database corruption. Some of the scenarios described can cause corruption, so you should regularly run consistency checks such as dbcc checkalloc and dbcc checkdb on all databases. See Chapter 26, “Checking Database Consistency,” for a discussion of these commands.

Disk mirroring commands

The disk mirror, disk unmirror, and disk remirror commands control disk mirroring. All the commands can be issued while the devices are in use, so you can start or stop database device mirroring while databases are being used.

Note The disk mirror, disk unmirror, and disk remirror commands alter the sysdevices table in the master database. After issuing any of these commands, you should dump the master database to ensure recovery in case master is damaged.

Initializing mirrors

disk mirror starts disk mirroring. *Do not* initialize the mirror device with disk init. A database device and its mirror constitute one logical device. The disk mirror command adds the mirror name to the mirrorname column in the sysdevices table.

Note To retain use of asynchronous I/O, always mirror devices that are capable of asynchronous I/O to other devices capable of asynchronous I/O. In most cases, this means mirroring raw devices to raw devices and operating system files to operating system files.

If the operating system cannot perform asynchronous I/O on files, mirroring a raw device to a regular file produces an error message. Mirroring a regular file to a raw device will work, but will not use asynchronous I/O.

The disk mirror syntax is:

```
disk mirror
    name = "device_name" ,
    mirror = "physicalname"
    [ , writes = { serial | noserial }]
```

The *device_name* is the name of the device that you want to mirror, as it is recorded in sysdevices.name (by disk init). Use the mirror = “*physicalname*” clause to specify the path to the mirror device, enclosed in single or double quotes. If the mirror device is a file, “*physicalname*” must unambiguously identify the path where Adaptive Server will create the file; it cannot specify the name of an existing file.

On systems that support asynchronous I/O, the writes option allows you to specify whether writes to the first device must finish before writes to the second device begin (serial) or whether both I/O requests are to be queued immediately, one to each side of the mirror (noserial). In either case, if a write cannot be completed, the I/O error causes the bad device to become unmirrored.

serial writes are the default. The writes to the devices take place consecutively, that is, the first one finishes before the second one starts. serial writes provide protection in the case of power failures: one write may be garbled, but both of them will not be. serial writes are generally slower than noserial writes.

In the following example, tranlog is the logical device name for a raw device. The tranlog device was initialized with disk init and is being used as a transaction log device (as in create database...log on *tranlog*). The following command mirrors the transaction log device:

```
disk mirror
  name = "tranlog",
  mirror = "/dev/rxy1e"
```

Unmirroring a device

Disk mirroring is automatically deactivated when one of the two physical devices fails. When a read or write to a mirrored device is unsuccessful, Adaptive Server prints error messages. Adaptive Server continues to run, unmirrored. You must remirror the disk to restart mirroring.

Use the disk unmirror command to stop the mirroring process during hardware maintenance:

```
disk unmirror
   name = "device_name"
   [, side = { "primary" | secondary }]
   [, mode = { retain | remove }]
```

The side option to the disk unmirror command allows you to specify which side of the mirror to disable. primary (in quotes) is the device listed in the name column of sysdevices; secondary (no quotes required) is the device listed in the mirrorname column of sysdevices. secondary is the default.

The mode option indicates whether the unmirroring process should be temporary (retain) or permanent (remove). retain is the default.

Temporarily deactivating a device

By default (mode=retain), Adaptive Server temporarily deactivates the specified device; you can reactivate it later. This is similar to what happens when a device fails and Adaptive Server activates its mirror:

- I/O is directed only at the remaining device of the mirrored pair.
- The status column of sysdevices is altered to indicate that the mirroring feature has been deactivated.
- The entries for primary (phyname) and secondary (mirrorname) disks are unchanged.

Permanently disabling a mirror

Use mode=remove to disable disk mirroring. This option eliminates all references in the system tables to a mirror device, but does *not* remove an operating system file that has been used as a mirror.

If you set mode=remove:

- The status column is altered to indicate that the mirroring feature is to be ignored.
- The phyname column is replaced by the name of the secondary device in the mirrorname column if the primary device is the one being deactivated.
- The mirrorname column is set to NULL.

Effects on system tables

The mode option changes the status column in sysdevices to indicate that mirroring has been disabled (see Table 16-2 on page 565). Its effects on the phyname and mirrorname columns in sysdevices depend on the side argument also, as shown in Table 17-1

Table 17-1: Effects of mode and side options to the disk mirror command

		side	
		primary	secondary
mode	remove	Name in mirrorname moved to phyname and mirrorname set to null; status changed	Name in mirrorname removed; status changed
	retain	Names unchanged; status changed to indicate which device is being deactivated	

This example suspends the operation of the primary device:

```
disk unmirror
  name = "tranlog",
  side = "primary"
```

Restarting mirrors

Use disk remirror to restart a mirror process that has been suspended due to a device failure or with disk unmirror. The syntax is:

```
disk remirror
   name = "device_name"
```

This command copies the database device to its mirror.

waitfor mirrorexit

Since disk failure can impair system security, you can include the waitfor mirrorexit command in an application to perform specific tasks when a disk becomes unmirrored:

```
begin
    waitfor mirrorexit
    commands to be executed
end
```

The commands depend on your applications. You may want to add certain warnings in applications that perform updates or use sp_dboption to make certain databases read-only if the disk becomes unmirrored.

Note Adaptive Server knows that a device has become unmirrored only when it attempts I/O to the mirror device. On mirrored databases, this occurs at a checkpoint or when the Adaptive Server buffer must be written to disk. On mirrored logs, I/O occurs when a process writes to the log, including any committed transaction that performs data modification, a checkpoint, or a database dump.

---

waitfor mirrorexit and the error messages that are printed to the console and error log on mirror failure are activated only by these events.

Mirroring the master device

If you choose to mirror the device that contains the master database, in a UNIX environment, you need to edit the runserver file for your Adaptive Server so that the mirror device starts when the server starts.

On UNIX, add the -r flag and the name of the mirror device:

```
dataserver -d /dev/rsd1f -r /dev/rs0e -e/sybase/install/errorlog
```

For information about mirroring the master device on Windows NT, see the *Utility Guide*.

Getting information about devices and mirrors

For a report on all Adaptive Server devices on your system (user database devices and their mirrors, as well as dump devices), execute sp_helpdevice.

Disk mirroring tutorial

The following steps illustrate the use of disk mirroring commands and their effect on selected columns of master..sysdevices. The status number and its hexidecimal equivalent for each entry in sysdevices are in parentheses:

Step 1

Initialize a new test device using:

```
disk init name = "test",
physname = "/usr/sybase/test.dat",
size=5120
```

This inserts the following values into columns of master..sysdevices:

```
name  phyname              mirrorname          status
test  /usr/sybase/test.dat   NULL                   16386
```

Status 16386 indicates that the device is a physical device (2, 0x00000002), and any writes are to a UNIX file (16384, 0x00004000). Since the mirrorname column is null, mirroring is not enabled on this device.

Step 2

Mirror the test device using:

```
disk mirror name = "test",
mirror = "/usr/sybase/test.mir"
```

This changes the master..sysdevices columns to:

```
name  phyname              mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

Step 3

Disable the mirror device (the secondary side), but retain that mirror:

```
disk unmirror name = "test",
side = secondary, mode = retain
```

```
name  phyname              mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  18658
```

Status 18658 indicates that the device is mirrored (64, 0x00000040), and the mirror device has been retained (2048, 0x00000800), but mirroring has been disabled (512 bit off), and only the primary device is used (256 bit off). Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file (16384, 0x00004000) and are in serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

Step 4

Remirror the test device:

```
disk remirror name = "test"
```

This resets the master..sysdevices columns to:

```
name   phyname                mirrorname           status
test   /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

Step 5

Disable the test device (the primary side), but retain that mirror:

```
disk unmirror name = "test",
side = "primary", mode = retain
```

This changes the master..sysdevices columns to:

```
name   phyname                mirrorname           status
test   /usr/sybase/test.dat   /usr/sybase/test.mir  16866
```

Status 16866 indicates that the device is mirrored (64, 0x00000040), but mirroring has been disabled (512 bit off) and that only the secondary device is used (256, 0x00000100). Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file (16384, 0x00004000), and are in serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

Step 6

Remirror the test device:

```
disk remirror name = "test"
```

This resets the master..sysdevices columns to:

```
name   phyname                mirrorname           status
test   /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

Status 17122 indicates that mirroring is currently enabled (512, 0x00000200) on this device. Reads are mirrored (128, 0x00000080), and writes are mirrored to a UNIX file device (16384, 0x00004000), the device is mirrored (64, 0x00000040), and serial (32, 0x00000020). The device is a physical disk (2, 0x00000002).

Step 7 Disable the test device (the primary side), and remove that mirror:

```
disk unmirror name = "test", side = "primary",
mode = remove
```

This changes the master..sysdevices columns to:

```
name  phyname               mirrorname          status
test  /usr/sybase/test.dat   NULL                       16386
```

Status 16386 indicates that the device is a physical device (2, 0x00000002), and any writes are to a UNIX file (16384, 0x00004000). Since the mirrorname column is null, mirroring is not enabled on this device.

Step 8 Remove the test device to complete the tutorial:

```
sp_dropdevice test
```

This removes all entries for the test device from master..sysdevices.

Disk resizing and mirroring

The disk resize command can be used only when mirroring is permanently disabled. If you try to run disk resize on a device that is mirrored, you see this:

```
disk resize can proceed only when mirroring is
permanently disabled. Unmirror secondary with mode =
'remove' and re-execute disk resize command.
```

When mirroring is only temporarily disabled, two scenarios can arise:

- The primary device is active while the secondary device is temporarily disabled and produces the same error as shown above.
- The secondary device is active while the primary device is temporarily disabled and produces an error with this message;

  ```
  disk resize can proceed only when mirroring is
  permanently disabled. Unmirror primary with mode
  ```

```
= 'remove' and re-execute the command.
```

v **Increasing the size of a mirrored device**

1. Permanently disable mirroring on the device.
2. Increase the size of the primary device.
3. Physically remove the mirror device (in case of file).
4. Reestablish mirroring.

CHAPTER 18 Configuring Memory

This chapter describes how Adaptive Server uses memory and explains how to maximize the memory available to Adaptive Server on your system.

Determining memory availability for Adaptive Server

The more memory that is available, the more resources Adaptive Server has for internal buffers and caches. Having enough memory available for caches reduces the number of times Adaptive Server has to read data or procedure plans from disk.

There is no performance penalty for configuring Adaptive Server to use the maximum amount of memory available on your computer. However, be sure to assess the other memory needs on your system first, and then configure the Adaptive Server to use only the remaining memory that is still available. Adaptive Server may not be able to start if it cannot acquire the memory for which it is configured.

To determine the maximum amount of memory available on your system for Adaptive Server:

1 Determine the total amount of physical memory on your computer system.

2 Subtract the memory required for the operating system from the total physical memory.

3 Subtract the memory required for Backup Server, Monitor Server, or other Adaptive Server-related software that must run on the same machine.

4 If the machine is not dedicated to Adaptive Server, also subtract the memory requirements for other system uses.

   For example, subtract the memory that will be used by any client applications that will run on the Adaptive Server machine. Windowing systems, such as X Windows, require a lot of memory and can interfere with Adaptive Server performance when used on the same machine as Adaptive Server.

The memory left over after subtracting requirements for the operating system and other applications is the total memory available for Adaptive Server. The value of the max memory configuration parameter specifies the maximum amount of memory to which Adaptive Server is configurable. See "Configuration parameters that affect memory allocation" on page 594 for information about Configure Adaptive Server to use this memory.

How Adaptive Server allocates memory

All database object pages are sized in terms of the **logical page size**, which you specify when you build a new master device. All databases – and all objects in every database– use the same logical page size. The size of Adaptive Server's logical pages (2, 4, 8, or 16K) determines the server's space allocation. Each allocation page, object allocation map (OAM) page, data page, index page, text page, and so on are built on a logical page. For example, if the logical page size of Adaptive Server is 8K, each of these page types are 8K in size. All of these pages consume the entire size specified by the size of the logical page. Larger logical pages allow you to create larger rows, which can improve your performance because Adaptive Server accesses more data each time it reads a page. For example, a 16K page can hold 8 times the amount of data as a 2K page, an 8K page holds 4 times as much data as a 2K page, and so on, for all the sizes for logical pages.

The logical page size is a server-wide setting; you cannot have databases with varying size logical pages within the same server. All tables are appropriately sized so that the row size is no greater than the current page size of the server. That is, rows cannot span multiple pages.

Regardless of the logical page size it is configured for, Adaptive Server allocates space for objects (tables, indexes, text page chains) in extents, each of which is eight logical pages. That is, if a server is configured for 2K logical pages, it allocates one extent, 16K, for each of these objects; if a server is configured for 16K logical pages, it allocates one extent, 128K, for each of these objects.

This is also true for system tables. If your server has many small tables, space consumption can be quite large if the server uses larger logical pages. For example, for a server configured for 2K logical pages, systypes—with approximately 31 short rows, a clustered and a non-clustered index—reserves 3 extents, or 48K of memory. If you migrate the server to use 8K pages, the space reserved for systypes is still 3 extents, 192K of memory. For a server configured for 16K, systypes requires 384K of disk space. For small tables, the space unused in the last extent can become significant on servers using larger logical page sizes.

Databases are also affected by larger page sizes. Each database includes the system catalogs and their indexes. If you migrate from a smaller to larger logical page size, you must account for the amount of disk space each database requires. Table 18-1 lists the minimum size for a database on each of the logical page sizes.

Table 18-1: Minimum database sizes

Logical page size	Minimum database size
2K	2MB
4K	4MB
8K	8MB
16K	16MB

Note that the logical page size is not the same as the memory allocation page size. Memory allocation page size is always 2K, regardless of logical page size, which can be 2, 4, 8, or 16K. Most memory related configure parameters use units of 2K for their memory page size. These configuration parameter include:

- max memory
- total logical memory

- total physical memory
- procedure cache size
- size of process object heap
- size of shared class heap
- size of global fixed heap

Disk space allocation

Note that the logical page size is not the same as the memory allocation page size. This is the unit in which disk space is allocated, and Adaptive Server allocated this space in 2K pages. Some of the configuration parameters use this 2K page size for their allocation units.

Larger logical page sizes and buffers

Adaptive Server allocates buffer pools in units of logical pages. For example, on a server using 2K logical pages, 8MB are allocated to the default data cache. This constitutes approximately 2048 buffers. If you allocated the same 8MB for the default data cache on a server using a 16K logical page size, the default data cache is approximately 256 buffers. On a busy system, this small number of buffers might result in a buffer always being in the wash region, causing a slowdown for tasks requesting clean buffers. In general, to obtain the same buffer management characteristics on larger page sizes as with 2K logical page sizes, you should scale the size of the caches to the larger page size. So, if you increase your logical page size by four times, your cache and pool sizes should be about four times larger as well.

Adaptive Server typically allocates memory dynamically and allocates memory for row processing as it needs it, allocating the maximum size for these buffers, even if large buffers are unnecessary. These memory management requests may cause Adaptive Server to have a marginal loss in performance when handling wide-character data.

Heap memory

A heap memory pool is an internal memory pool created at start-up that tasks use to dynamically allocate memory as needed. This memory pool is used by tasks that requires a lot of memory from the stack, such as tasks that use wide columns. For example, if you make a wide column or row change, the temporary buffer this task uses can be as larger as 16K, which is too big to allocate from the stack. Adaptive Server dynamically allocates and frees memory during the task's run time. The heap memory pool dramatically reduces the pre-declared stack size for each task while also improving the efficiency of memory usage in the server. The heap memory the task uses is returned to the heap memory pool when the task is finished.

Set the heap memory with the heap memory per user configuration parameter. The syntax for heap memory per user is:

sp_configure 'heap memory per user', *amount_of_memory*

Heap memory is measured in bytes per user. By default, the amount of memory is set to 4096 bytes. This example specifies setting the default amount of heap memory for 10 users:

```
sp_configure 'heap memory per user', 10
```

You can also specify the amount of memory in the number of bytes per user. For example, the following example specifies that each user connection is allocated 4K bytes of heap memory:

```
sp_configure 'heap memory per user', 0, "4K"
```

At the initial Adaptive Server configuration, 1MB is set aside for heap memory. Additional heap memory is allocated for all the user connections and worker processes for which the server is configured, so the following configuration parameters affect the amount of heap memory available when the server starts:

- number of user connections
- number of worker processes

The global variable *@@heapmemsize* reports the size of the heap memory pool, in bytes.

Calculating heap memory

To calculate how much heap memory Adaptive Server sets aside, perform the following (Adaptive Server reserves a small amount of memory for internal structures, so these numbers will vary from site to site):

((1024 * 1024) + (*heap memory in bytes*)* (number of user connections + number of worker processes))

The initial value of (1024 * 1024) is the 1MB initial size of the heap memory pool. Adaptive Server reserves a small amount of memory for internal structures.

For example, if your server is configured for:

- heap memory per user – 4K
- number of user connectins – 25 (the default)
- number of worker processes – 25 (the default)

@@heapmemsize reports 1378304 bytes.

And the estimated value using the formula above, is:

((1024 X 1024) + (4 * 1024 * 50)) = 1253376

Now, if you increase the number of user connections, the size of the heap memory pool increases accordingly:

```
sp_configure 'user connections', 100
```

@@heapmemsize reports 1716224 bytes.

The estimated value in this case comes out to be:

((1024 * 1024) + (4 * 1024 * (100 + 25)) = 1560576

If your applications were to fail with the following error message:

```
There is insufficient heap memory to allocate %ld
bytes. Please increase configuration parameter 'heap
memory per user' or try again when there is less
activity on the system.
```

You can increase the heap memory available to the server by increasing one of:

- heap memory per user
- number of user connections
- number of worker processes

Sybase recommends that you first try to increase the heap memory per user configuration option before you increase number of user connections or number of worker processes. Increasing the number of user connections and number of worker processes first consumes system memory for other resources, which may cause you to increase the server's max memory.

See Chapter 5, “Setting Configuration Parameters” in the *System Administration Guide* for more information on how to make memory related configuration option changes.

The size of the memory pool depends on the number of user connections. Sybase recommends that you set heap memory per user to at least three times the size of your logical page.

How Adaptive Server uses memory

Memory exists in Adaptive Server as total logical or physical memory:

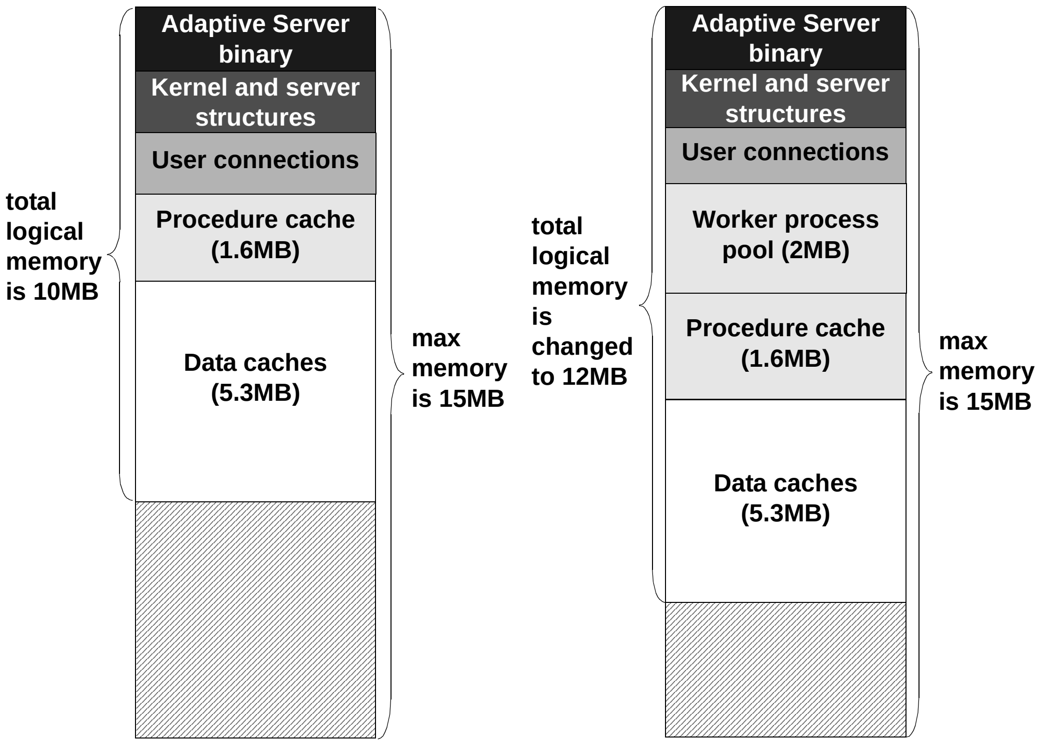
- Total logical memory – is the sum of the memory required for all the sp_configure parameters. The total logical memory is required to be available, but may or may not be in use at a given moment. The total logical memory value may change due to changes in the configuration parameter values.
- Total physical memory – is the sum of all shared memory segments in Adaptive Server. That is, total physical memory is the amount of memory Adaptive Server uses at a given moment. You can verify this value with the read-only configuration parameter total physical memory. The value of total physical memory can only increase because Adaptive Server does not shrink memory pools once they are allocated. You can decrease the amount of total physical memory by changing the configuration parameters and restarting Adaptive Server.

When Adaptive Server starts, it allocates memory for:

- Memory used by Adaptive Server for nonconfigurable data structures
- Memory for all user-configurable parameters, including the data cache, the procedure cache, and the default data cache.

Figure 18-1 illustrates how Adaptive Server allocates memory as you change some of the memory configuration parameters:

Figure 18-1: How Adaptive Server handles memory configuration changes



When a 2MB worker process pool is added to the Adaptive Server memory configuration, the procedure and data caches maintain their originally configured sizes; 1.6MB and 5.3MB, respectively. Because max memory is 5MB larger than the total logical memory size, it easily absorbs the added memory pool. If the new worker process pool brings the size of the server above the limit of max memory, any command you issue to increase the worker process pool fails. If this happens, the total logical memory required for the new configuration is indicated in the sp_configure failure message. Set the value of max memory to a value greater than the total logical memory required by the new configuration. Then retry your sp_configure request.

The size of the default data cache and the procedure cache has a significant impact on overall performance. See Chapter 10, “Memory Use and Performance,” in *Performance and Tuning: Basics* for recommendations on optimizing procedure cache size.

How much memory does Adaptive Server need?

The total memory Adaptive Server requires to start is the *sum of all memory configuration parameters* plus the *size of the procedure cache* plus the *size of the buffer cache,* where the size of the procedure cache and the size of the buffer cache are expressed in round numbers rather than in percentages. The procedure cache size and buffer cache size do *not* depend on the total memory you configure. You can configure the procedure cache size and buffer cache size independently. Use sp_cacheconfig to obtain information such as the total size of each cache, the number of pools for each cache, the size of each pool, and so on.

Use sp_configure to determine the total amount of memory Adaptive Server is using at a given moment. For example:

```
1> sp_configure "total logical memory"

Parameter Name        Default   Memory Used   Config Value   Run Value
   Unit               Type
--------------        -------------- --- ----------- ----------- ---------
------------------- ----------
total logical memory  33792    127550         63775         63775
   memory pages(2k)   read-only
```

The value for the Memory Used column is represented in kilobytes, while the value for the Config Value column is represented in 2K pages.

The config value indicates the total logical memory Adaptive Server uses while it is running. The run value column shows the total logical memory being consumed by the current Adaptive Server configuration. You will see a different output if you run this example because no two Adaptive Servers are likely to be configured in exactly the same fashion.

For more information about sp_configure please see *Sybase Adaptive Server Enterprise Reference Manual Volume 3: Procedures.*

If you are upgrading

If you upgrade to release 12.5 Adaptive Server or higher, pre-12.5 Adaptive Server configuration values for total logical memory, procedure cache percent, and min online engines are used to calculate the new values for procedure cache size and number of engines at startup. Adaptive Server computes the size of the default data cache during the upgrade and writes this value to the configuration file. If the computed sizes of the data cache or procedure cache are less than the default sizes, they are reset to the default. During the upgrade, max memory is set to the value of total logical memory specified in the configuration file.

You should reset the value of max memory to comply with the resource requirements.

You can use the verify option of sp_configure to verify any changes you make to the configuration file without having to restart Adaptive Server. The syntax is:

```
sp_configure “configuration file”, 0, “verify”, “full_path_to_file”
```

Configuration parameters that affect memory allocation

When setting Adaptive Server’s memory configuration, you specify each memory requirement with an absolute value, using sp_configure. You also specify the size of the procedure and default data caches in an absolute value.

There are three configuration parameters that affect the way in which memory is allocated. They are:

max memory

The configuration parameter max memory allows you to establish a maximum setting for the amount of memory you can allocate to Adaptive Server. Setting max memory to a slightly larger value than immediately necessary provides extra memory that is utilized when Adaptive Server’s memory needs are increased.

The way Adaptive Server allocates the memory specified by max memory depends on how you configure allocate max shared memory and dynamic allocation on demand.

allocate max shared memory

The allocate max shared memory parameter allows you to either allocate all the memory specified by max memory at start-up or to allocate only the memory required by the total logical memory specification during start-up.

On some platforms, if the number of shared memory segments allocated to an application is more than an optimal, platform-specific number, it could result in performance degradation. If this occurs, set max memory to the maximum amount available for Adaptive Server. Set allocate max shared memory to 1 and restart the server. This ensures that all the memory for max memory will be allocated by Adaptive Server during start time with the least number of segments.

For example, if you set allocate max shared memory to 0 (the default) and max memory to 500MB, but the server configuration only requires 100MB of memory at start-up, Adaptive Server allocates the remaining 400MB only when it requires the additional memory. However, if you set allocate max shared memory to 1, Adaptive Server allocates the entire 500MB when it starts.

The advantage of allocating all the memory at start-up, by setting allocate max shared memory to 1, is that there is no performance degradation while the server is readjusting for the additional memory. However, if you do not predict memory growth properly, and max memory is set to a large value, you may be wasting physical memory. Since you cannot dynamically decrease memory configuration parameters, it is important that you take into account other memory requirements.

dynamic allocation on demand

dynamic allocation on demand allows you to determine whether your memory resources are allocated as soon as they are requested or only as they are needed. Setting dynamic allocation on demand to 1 allocates memory changes as needed, and setting it to 0 allocates the configured memory requested in the memory configuration change at the time of the memory reconfiguration.

For example, if you set the value of dynamic allocation on demand to 1 and you change number of user connections to 1024, the total logical memory is 1024 multiplied by the amount of memory per user. If the amount of memory per user is 112K, then the memory for user connections is 112MB (1024 x 112).

This is the maximum amount of memory that the number of user connections configuration parameter is allowed to use. However, if only 500 users are connected to the server, the amount of total physical memory used by the number of user connections parameter is 56MB (500 x 112).

Now assume the value of dynamic allocation on demand is 0; when you change number of user connections to 1024, all user connection resources are configured immediately.

Optimally, you should organize Adaptive Server's memory so that the amount of total physical memory is less than the amount of total logical memory, which is less than the max memory. This can be achieved, in part, by setting the value of dynamic allocation on demand to 1, and setting the value of allocate max shared memory to 0.

Dynamically allocating memory

Adaptive Server allocates physical memory dynamically. This allows users to change the memory configuration of Adaptive Server without restarting the server.

Note Adaptive Server allocates memory dynamically; however, it does not decrease memory dynamically. It is important that you accurately assess the needs of your system, because you may need to restart the server if you decrease the memory configuration parameters and want to release previously used physical memory. See "Dynamically decreasing memory configuration parameters" on page 597 for more information.

Consider changing the value of the max_memory configuration parameter:

- When you change the amount of RAM on your machine
- When the pattern of use of your machine changes
- When the configuration fails because max_memory is insufficient.

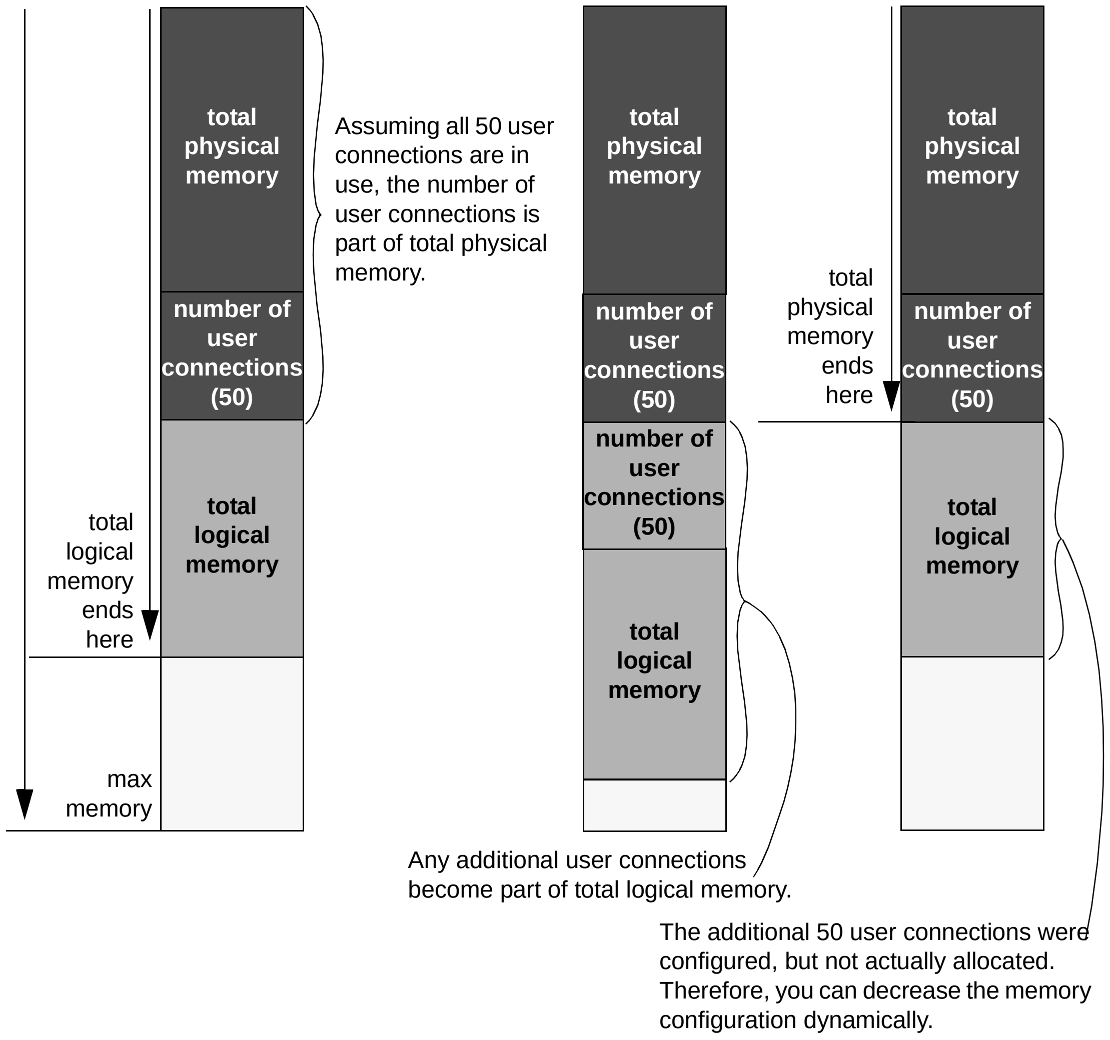
If Adaptive Server cannot start

When Adaptive Server starts, it must acquire the full amount of memory set by total logical memory from the operating system. If Adaptive Server cannot start because it cannot acquire enough memory, reduce the memory requirements by reducing the values for the configuration parameters that consume memory. You may also need to reduce the values for other configuration parameters that require large amounts of memory. Then restart Adaptive Server to use the new values. See Chapter 4, “Setting Configuration Parameters,” for information about using configuration files.

Dynamically decreasing memory configuration parameters

If you reset memory configuration parameters to a lower value, any engaged memory will not be released dynamically. To see how the changes in memory configuration are decreased, see Figure 18-2 and Figure 18-3.

Figure 18-2: dynamic allocation on demand set to 1 with no new user connections


total physical memory
number of user connections (50)
total logical memory
total logical memory ends here
max memory
Assuming all 50 user connections are in use, the number of user connections is part of total physical memory.
total physical memory
number of user connections (50)
number of user connections (50)
total logical memory
Any additional user connections become part of total logical memory.
total physical memory ends here
total physical memory
number of user connections (50)
total logical memory
The additional 50 user connections were configured, but not actually allocated. Therefore, you can decrease the memory configuration dynamically.

In Figure 18-2, because dynamic allocation on demand is set to 1, memory is now used only when there is an event that triggers a need for additional memory use. In this example, such an event would be a request for additional user connections, when a client attempts to log into Adaptive Server.

You may decrease number of user connections to a number that is greater than or equal to the number of user connections actually allocated, because, with dynamic allocation on demand set to 1, and without an actual increase in user connection request, no additional memory is required from the server.

Figure 18-3: dynamic allocation on demand set to 1, with new user connections logged on

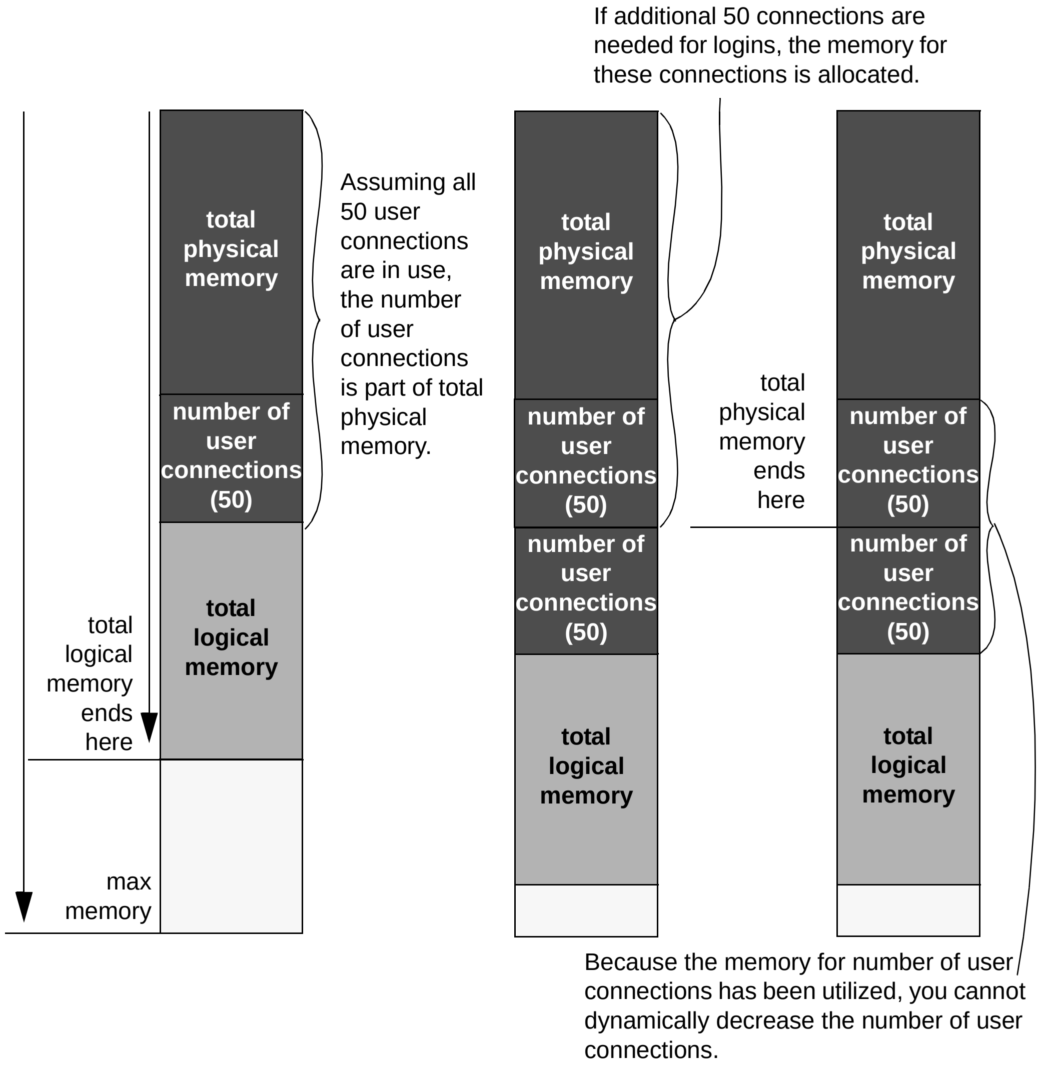
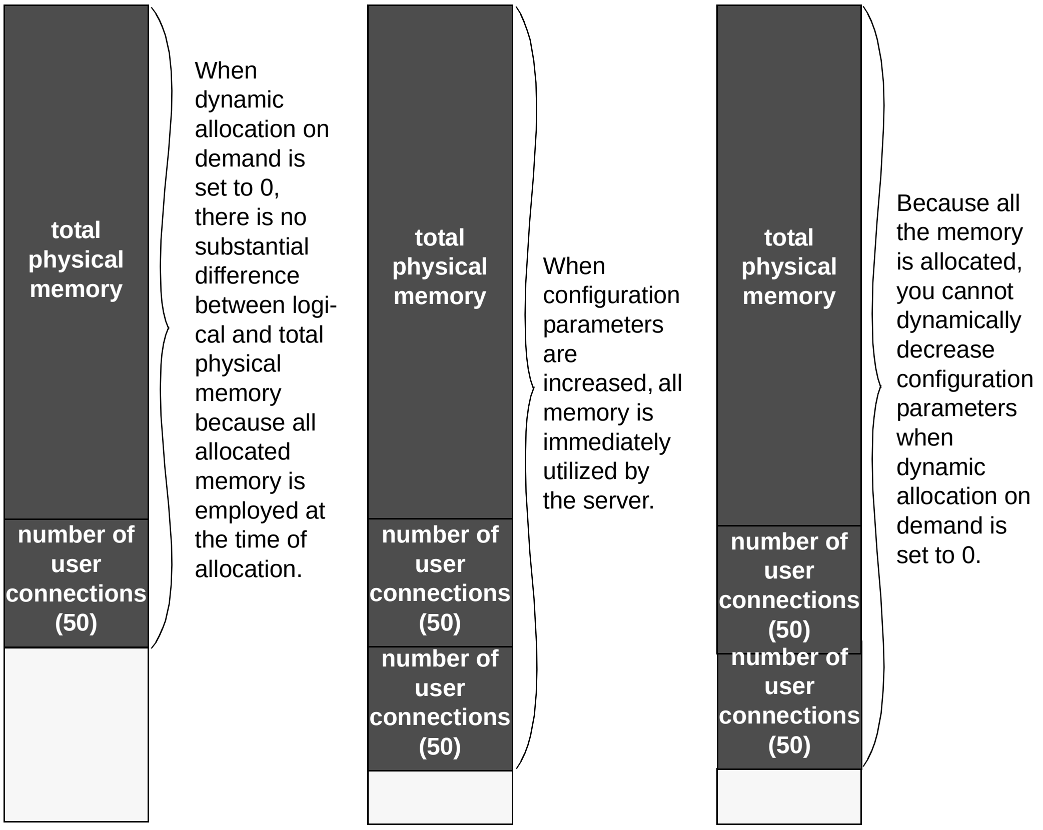

If additional 50 connections are needed for logins, the memory for these connections is allocated.
total physical memory
number of user connections (50)
total logical memory
total logical memory ends here
max memory
Assuming all 50 user connections are in use, the number of user connections is part of total physical memory.
total physical memory
number of user connections (50)
number of user connections (50)
total logical memory
total physical memory ends here
total physical memory
number of user connections (50)
number of user connections (50)
total logical memory
Because the memory for number of user connections has been utilized, you cannot dynamically decrease the number of user connections.

Figure 18-3 assumes that each of the additional 50 user connections is actually used. You cannot decrease number of user connections, because the memory is in use. You can use sp_configure to specify a change to memory configuration parameters, but this change will not take place until the server is restarted.

Figure 18-4: dynamic allocation on demand set to 0


total physical memory
number of user connections (50)
When dynamic allocation on demand is set to 0, there is no substantial difference between logical and total physical memory because all allocated memory is employed at the time of allocation.
total physical memory
number of user connections (50)
number of user connections (50)
When configuration parameters are increased, all memory is immediately utilized by the server.
total physical memory
number of user connections (50)
number of user connections (50)
Because all the memory is allocated, you cannot dynamically decrease configuration parameters when dynamic allocation on demand is set to 0.

Note In theory, when dynamic allocation on demand is set to 0, there should be no difference between total logical and physical memory. However, there are some discrepancies in the way that Adaptive Server estimates memory needs, and the way in which memory is actually required for usage. For this reason, you may see a difference between the two during runtime.

When dynamic allocation on demand is set to 0, all configured memory requirements are immediately allocated. You cannot dynamically decrease memory configuration.

In Figure 18-3 and Figure 18-4, users can change the memory configuration parameter values to any smaller, valid value. While this change does not take place dynamically, it disallows new memory use. For example, if you have configured number of user connections to allow for 100 user connections and then change that value to 50 user connections, in the situations represented by Figure 18-3 and Figure 18-4 you can decrease the number of user connections value back to 50. This change does not effect the memory used by Adaptive Server until after the server is restarted, but it prevents any new users from logging onto the server.

System procedures for configuring memory

The three system procedures that you need to use while configuring Adaptive Server memory are:

- sp_configure
- sp_helpconfig
- sp_monitorconfig

Using *sp_configure* to set configuration parameters

The full syntax and usage of sp_configure and details on each configuration parameter are covered in Chapter 4, “Setting Configuration Parameters.” The rest of this chapter discusses issues pertinent to configuring the parameters that use Adaptive Server memory.

Execute sp_configure, specifying “Memory Use,” to see these parameter settings on your server.

```
sp_configure "Memory Use"
```

A “#” in the “Memory Used” column indicates that this parameter is a component of another parameter and that its memory use is included in the memory use for the other component. For example, memory used for stack size and stack guard size contributes to the memory requirements for each user connection and worker process, so the value is included in the memory required for number of user connections and for number of worker processes, if this is more than 200.

Some of the values in this list are computed values. They cannot be set directly with sp_configure, but are reported to show where memory is allocated. Among the computed values is total data cache size.

How much memory is available for dynamic growth?

Issuing sp_configure memory displays all of the memory parameters and determines the difference between max memory and total logical memory, which is the amount of memory available for dynamic growth. For example:

```
1> sp_configure memory

Msg 17411, Level 16, State 1:
Procedure 'sp_configure', Line 187:
Configuration option is not unique.
Parameter Name                 Default  Memory Used  Config Value  Run Value
Unit                 Type
------------------------------ ----------- ----------- ------------ ----------
-------------------- ----------
additional network memory            0            0             0           0
bytes                dynamic
allocate max shared memory           0            0             0           0
switch               dynamic
heap memory per user              4096            0          4096        4096
bytes                dynamic
lock shared memory                   0            0             0           0
switch               static
max memory                       33792       300000        150000      150000
memory pages(2k)    dynamic
memory alignment boundary        16384            0         16384       16384
bytes               static
memory per worker process         1024            4          1024        1024
bytes               dynamic
shared memory starting address       0            0             0           0
not applicable      static
total logical memory             33792       110994         55497       55497
memory pages(2k)    read-only
```

```
total physical memory                0        97656            0       48828
memory pages(2k)    read-only

An additional 189006 K bytes of memory is available for reconfiguration. This is
the difference between 'max memory' and 'total logical memory'.
```

Using *sp_helpconfig* to get help on configuration parameters

sp_helpconfig estimates the amount of memory required for a given configuration parameter and value. It also provides a short description of the parameter, information about the minimum, maximum, and default values for the parameter, the run value, and the amount of memory used at the current run value. sp_helpconfig is particularly useful if you are planning substantial changes to a server, such as loading large, existing databases from other servers, and you want to estimate how much memory is needed.

To see how much memory is required to configure a parameter, enter enough of the parameter name so that it is a unique name and the value you want to configure:

```
1> sp_helpconfig "worker processes", "50"
```

```
number of worker processes is the maximum number of worker processes that can be
in use Server-wide at any one time.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
0              2147483647     0              0              0
number   dynamic

Configuration parameter, 'number of worker processes', will consume 7091K of
memory if configured at 50.
Changing the value of 'number of worker processes' to '50' increases the amount
of memory ASE uses by 7178 K.
```

You can also use sp_helpconfig to determine the value to use for sp_configure, if you know how much memory you want to allocate to a specific resource:

```
1> sp_helpconfig "user connections", "5M"
```

```
number of user connections sets the maximum number of user connections that can
be connected to SQL Server at one time.
```

```
Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit      Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
5              2147483647     25             25             3773
number   dynamic
Configuration parameter, 'number of user connections', can be configured to 33
to fit in 5M of memory.
```

The important difference between the syntax of these two statements is the use of a unit of measure in the second example to indicate to the procedure that the value is a size, not a configuration value. The valid units of measure are:

- P – pages, (Adaptive Server 2K pages)
- K – kilobytes
- M – megabytes
- G – gigabytes

There are some cases where the syntax does not make sense for the type of parameter, or where Adaptive Server is not able to calculate the memory use. sp_helpconfig prints an error message in these cases. For example if you attempt to specify a size for a parameter that toggles, such as allow resource limits, sp_helpconfig prints the message that describes the function of the parameter for all the configuration parameters that do not use memory.

Using *sp_monitorconfig* to find metadata cache usage statistics

sp_monitorconfig displays metadata cache usage statistics on certain shared server resources, including:

- The number of databases, objects, and indexes that can be open at any one time
- The number of auxiliary scan descriptors used by referential integrity queries
- The number of free and active descriptors
- The percentage of active descriptors
- The maximum number of descriptors used since the server was last started

- The current size of the procedure cache and the amount actually used.

For example, suppose you have configured the number of open indexes configuration parameter to 500. During a peak period, you can run sp_monitorconfig as follows to get an accurate reading of the actual metadata cache usage for index descriptors. For example:

```
1> sp_monitorconfig "number of open indexes"

Usage information at date and time: Apr 22 2002  2:49PM.
Name             num_free  num_active  pct_act  Max_Used  Reused
--------------   --------  ----------  -------  --------  ------
number of open        217         283    56.60       300  No
```

In this report, the maximum number of open indexes used since the server was last started is 300, even though Adaptive Server is configured for 500. Therefore, you can reset the number of open indexes configuration parameter to 330, to accommodate the 300 maximum used index descriptors, plus space for 10 percent more.

You can also determine the current size of the procedure cache with sp_monitorconfig procedure cache size. This parameter describes the amount of space in the procedure cache is currently configured for and the most it has ever actually used. For example, the procedure cache in the following server is configured for 20,000 pages:

```
1> sp_configure "procedure cache size"

option_name                    config_value run_value
------------------------------ ------------ ---------
procedure cache size                   3271      3271
```

However, when you run sp_montorconfig “procedure cache size”, you find that the most the procedure cache has ever used is 14241 pages, which means that you can lower the run value of the procedure cache, saving memory:

```
1> sp_monitorconfig "procedure cache size"

Usage information at date and time: Apr 22 2002  2:49PM.
Name                num_free  num_active  pct_act  Max_Used  Reused
--------------      --------  ----------  -------  --------  ------
procedure cache         5878       14122    70.61     14241  No
```

Major uses of Adaptive Server memory

This section discusses configuration parameters that use large amounts of Adaptive Server memory and those that are commonly changed at a large number of Adaptive Server installations. These parameters should be checked by System Administrators who are configuring an Adaptive Server for the first time. System Administrators should review these parameters when the system configuration changes, after upgrading to a new versionof Adaptive Server, or when making changes to other configuration variables that use memory.

Configuration parameters that use less memory, or that are less frequently used, are discussed in "Other parameters that use memory" on page 611.

Adaptive Server executable code size

The size of the executable code is not included in the value of total logical memory or max memory calculations. total logical memory reports the actual memory requirement for Adaptive Server configuration excluding any memory required for the executable.

The memory required for the executable is managed by the operating system and is beyond the control of Adaptive Server. Please consult your operating system documentation for more details.

Data and procedure caches

As explained in "How Adaptive Server uses memory" on page 591, you specify the size of the data and procedure caches. Having sufficient data and procedure cache space is one of the most significant contributors to performance. This section explains the details between the two caches and how to monitor cache sizes.

Determining the procedure cache size

procedure cache size specifies the size of your procedure cache in 2K pages, regardless of the server's logical page size. For example:

```
1> sp_configure "procedure cache size"

Parameter Name        Default  Memory Used  Config Value  Run Value
```

```
Unit                Type
--------------------  -------  -----------  ------------  ---------
---------------  ---------
procedure cache size     3271         6914          3271      3271
memory pages(2k)   dynamic
```

The amount of memory used for the procedure cache is 8.248MB. To set the procedure cache to a different size, issue the following:

```
sp_configure "procedure cache size", new_size
```

This example resets the procedure cache size to 10000 2K pages (20MB):

```
sp_configure "procedure cache size", 10000
```

Determining the default data cache size

Both sp_cacheconfig and sp_helpcache display the current default data cache in megabytes. For example, the following shows an Adaptive Server configured with 19.86MB of default data cache:

```
sp_cacheconfig


Cache Name                Status     Type       Config Value    Run Value
------------------        --------   --------   ------------    ----------
default data cache        Active     Default    0.00 Mb         19.86Mb
                                                ------------    --------
                        Total                   0.00Mb          19.86 Mb
========================================================================
Cache: default data cache, Status: Active, Type: Default
       Config Size: 0.00 Mb, Run Size: 19.86 Mb
       Config Replacement: strict LRU, Run Replacement: strict LRU
       Config Partition:    1,   Run Partition:    1
IO Size       Wash Size     Config Size     Run Size      APF Percent
--------      ---------     ------------    ----------    -----------
2 Kb          4066 Kb        0.00 Mb         19.86 Mb               10
```

To change the default data cache, issue sp_cacheconfig, and specify “default data cache.” For example, to change the default data cache to 25MB, enter:

```
sp_cacheconfig "default data cache", "25M"
```

This change is dymamic.

The default data cache size is an absolute value, and the minimum size for the cache size is 256 times the logical page size; for 2K, the minimum is 512K, for 16K, the minimum is 4M. The default value is 8MB. During the upgrade process, Adaptive Server sets the default data cache size to the value of the default data cache in the configuration file.

Monitoring cache space

You can check data cache and procedure cache space with sp_configure:

```
sp_configure "total data cache size"
```

Monitoring cache sizes using the errorlog

Another way to determine how Adaptive Server uses memory is to examine the memory-related messages written to the error log when Adaptive Server starts. These messages state exactly how much data and procedure cache is allocated, how many **compiled objects** can reside in cache at any one time, and buffer pool sizes.

These messages provide the most accurate information regarding cache allocation on Adaptive Server. As discussed earlier, the amount of memory allocated for the procedure caches depends on the run value of the procedure cache size configuration parameter.

Each of these error log messages is described below.

Procedure cache messages

Two error log messages provide information about the procedure cache.

```
server: Number of proc buffers allocated: 556
```

This message states the total number of procedure buffers (proc buffers) allocated in the procedure cache.

```
server: Number of blocks left for proc headers: 629
```

This message indicates the total number of procedure headers (proc headers) available for use in the procedure cache.

proc buffer

A *proc buffer* (procedure buffer) is a data structure used to manage compiled objects in the procedure cache. One proc buffer is used for every copy of a compiled object stored in the procedure cache. When Adaptive Server starts, it determines the number of proc buffers required and multiplies that value by the size of a single proc buffer (76 bytes) to obtain the total amount of memory required.

proc header

A *proc header* (procedure header) is where a compiled object is stored while in the procedure cache. Depending on the size of the object to be stored, one or more proc headers may be required. The total number of compiled objects that can be stored in the procedure cache is limited by the number of available proc headers or proc buffers, whichever is less.

The total size of procedure cache is the combined total of memory allocated to proc buffers (rounded up to the nearest page boundary), plus the memory allocated to proc headers.

Data cache messages

When Adaptive Server starts, it records the total size of each cache and the size of each pool in the cache in the error log. This example shows the default data cache with two pools and a user-defined cache with two pools:

```
Memory allocated for the default data cache cache: 8030 Kb
Size of the 2K memory pool: 7006 Kb
Size of the 16K memory pool: 1024 Kb
Memory allocated for the tuncache cache: 1024 Kb
Size of the 2K memory pool: 512 Kb
Size of the 16K memory pool: 512 Kb
```

User connections

The amount of memory required per user connection varies by platform, and it changes when you change other configuration variables, including:

- default network packet size
- stack size and stack guard size
- user log cache size

Changing any of these parameters changes the amount of space used by each user connection: you have to multiply the difference in size by the number of user connections. For example, if you have 300 user connections, and you are considering increasing the stack size from 34K to 40K, the new value requires 1800K more memory.

Open databases, open indexes, and open objects

The three configuration parameters that control the total number of databases, indexes, and objects that can be open at one time are managed by special caches called **metadata caches**. The metadata caches reside in the kernel and server structures portion of Adaptive Server memory. You configure space for each of these caches with these parameters:

- number of open databases
- number of open indexes
- number of open objects

When Adaptive Server opens a database or accesses an index or an object, it needs to read information about it in the corresponding system tables: sysdatabases, sysindexes, and sysobjects. The metadata caches for databases, indexes, or objects let Adaptive Server access the information that describes it in the sysdatabases, sysindexes, or sysobjects row directly in its in-memory structure. This improves performance because Adaptive Server bypasses expensive calls that require disk access. It also reduces synchronization and spinlock contention when Adaptive Server has to retrieve database, index, or object information at runtime.

Managing individual metadata caches for databases, indexes, or objects is beneficial for a database that contains a large number of indexes and objects and where there is high concurrency among users. For more information about configuring the number of metadata caches, see “number of open databases” on page 117, “number of open indexes” on page 119, and “number of open objects” on page 120.

Number of locks

All processes in Adaptive Server share a pool of lock structures. As a first estimate for configuring the number of locks, multiply the number of concurrent user connections you expect, *plus* the number of worker processes that you have configured, by 20. The number of locks required by queries can vary widely. See “number of locks” on page 109 for more information. For information on how worker processes use memory, see “Worker processes” on page 612.

Database devices and disk I/O structures

The number of devices configuration parameter controls the number of database devices that can be used by Adaptive Server for storing data. See “number of devices” on page 86 for more information.

When a user process needs to perform a physical I/O, the I/O is queued in a disk I/O structure. See “disk i/o structures” on page 86 for more information.

Other parameters that use memory

This section discusses configuration parameters that use moderate amounts of memory or are infrequently used.

Parallel processing

Parallel processing requires more memory than serial processing. The configuration parameters that affect parallel processing are:

- number of worker processes
- memory per worker process
- partition groups
- number of mailboxes and number of messages

Worker processes

The configuration parameter number of worker processes sets the total number of worker processes available at one time in Adaptive Server. Each worker process requires about the same amount of memory as a user connection.

Changing any of these parameters changes the amount of memory required for each worker process:

- default network packet size
- stack size and stack guard size
- user log cache size
- memory per worker process

The memory per worker process configuration parameter controls the additional memory that is placed in a pool for all worker processes. This additional memory stores miscellaneous data structure overhead and inter-worker process communication buffers. See the *Performance and Tuning Guide* for information on setting memory per worker process.

Parallel queries and the procedure cache

Each worker process makes its own copy of the query plan in space borrowed from the procedure cache. The coordinating process keeps two copies of the query plan in memory.

Partition groups

You need to reconfigure the value only if you use a very large number of partitions in the tables on your server. See "partition groups" on page 173 for more information.

Remote servers

Some configuration parameters that allow Adaptive Server to communicate with other Sybase servers such as Backup Server, Component Integration Services, or XP Server use memory.

The configuration parameters that affect remote servers and that use memory are:

- number of remote sites

- number of remote connections
- number of remote logins
- remote server pre-read packets

Number of remote sites

Set the number of remote sites configuration parameter to the number of simultaneous sites you need to communicate to or from on your server. If you use only Backup Server, and no other remote servers, you can increase your data cache and procedure cache space by reducing this parameter to 1.

The connection from Adaptive Server to XP Server uses one remote site.

Other configuration parameters for RPCs

These configuration parameters for remote communication use only a small amount of memory for each connection:

- number of remote connections
- number of remote logins

Each simultaneous connection from Adaptive Server to XP Server for ESP execution uses one remote connection and one remote login.

Since the remote server pre-read packets parameter increases the space required for each connection configured by the number of remote connections parameter, increasing the number of pre-read packets can have a significant effect on memory use.

Referential integrity

If the tables in your databases use a large number of referential constraints, you may need to adjust the number of aux scan descriptors parameter, if user connections exceed the default setting. In most cases, the default setting is sufficient. If a user connection exceeds the current setting, Adaptive Server returns an error message suggesting that you increase the number of aux scan descriptors parameter setting.

Other parameters that affect memory

Other parameters that affect memory are listed below. When you reset these configuration parameters, check the amount of memory they use and the effects of the change on your procedure and data cache.

• additional network memory	• max SQL text monitored
• allow resource limits	• number of alarms
• audit queue size	• number of large i/o buffers
• event buffers per engine	• permission cache entries
• max number network listeners	
• max online engines	

CHAPTER 19 Configuring Data Caches

This chapter describes how to create and administer named caches on Adaptive Server.

The most common reason for administering data caches is to reconfigure them for performance. This chapter is primarily concerned with the mechanics of working with data caches. Chapter 10, “Memory Use and Performance,” in the *Performance and Tuning Guide: Basics* discusses performance concepts associated with data caches.

Adaptive Server's data cache

The data cache holds the data, index, and log pages currently in use as well as pages used recently by Adaptive Server. When you first install Adaptive Server, it has a single default data cache that is used for all data, index, and log activity. The default size of this cache is 8M. Creating other caches does not reduce the size of the default data cache. Also, you can create pools within the named caches and the default cache to perform large I/Os. You can then bind a database, table (including the syslogs table), index, or text or image page chain to a named data cache.

Large I/O sizes enable Adaptive Server to perform data prefetching when the query optimizer determines that prefetching would improve performance. For example, an I/O size of 128K on a server configured with 16K logical pages means that Adaptive Server can read an entire extent—8 pages—all at once, rather than performing 8 separate I/Os.

Sorts can also take advantage of buffer pools configured for large I/O sizes.

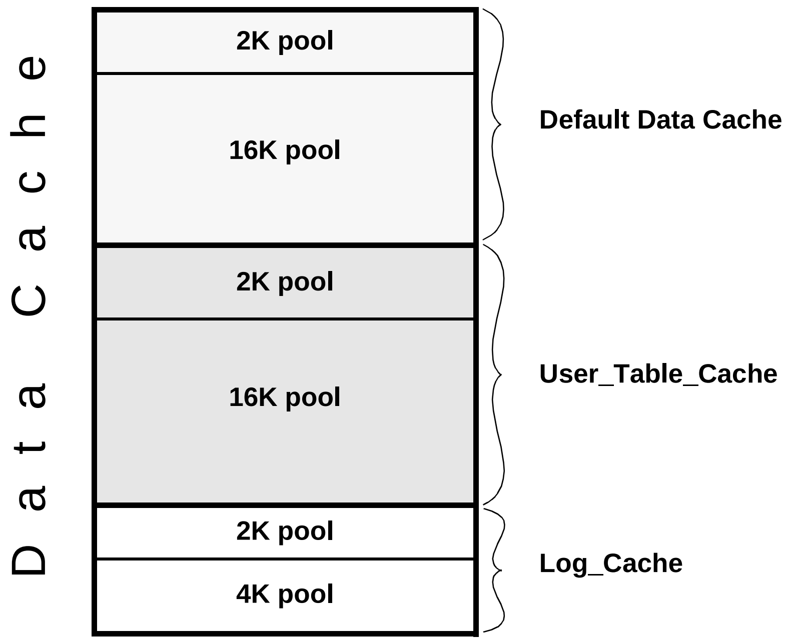
Configuring named data caches does not divide the default cache into separate cache structures. The named data caches that you create can be used only by databases or database objects that are explicitly bound to them. All objects not explicitly bound to named data caches use the default data cache.

Adaptive Server provides user-configurable data caches to improve performance, especially for multiprocessor servers. For information about how the data cache affects performance, see "Data cache" of the *Performance and Tuning: Basics*.

Figure 19-1 shows a cache with the default data cache and two named data caches. This server uses 2K logical pages.

The default cache contains a 2K pool and a 16K pool. The User_Table_Cache cache also has a 2K pool and a 16K pool. The Log_Cache has a 2K pool and a 4K pool.

Figure 19-1: Data cache with default cache and two named data caches



Cache configuration commands

The following table lists commands for configuring named data caches, for binding and unbinding objects to caches, and for reporting on cache bindings. It also lists procedures you can use to check the size of your database objects, and commands that control cache usage at the object, command, or session level.

Command	Function
sp_cacheconfig	Creates or drops named caches, and changes the size, cache type, cache policy, or number of cache partitions.
sp_poolconfig	Creates and drops I/O pools, and changes their size, wash size, and asynchronous prefetch percent limit.
sp_bindcache	Binds databases or database objects to a cache.
sp_unbindcache	Unbinds specific objects or databases from a cache.
sp_unbindcache_all	Unbinds all objects bound to a specified cache.

Command	**Function**
sp_helpcache	Reports summary information about data caches and lists the databases and database objects that are bound to caches.
sp_cachestrategy	Reports on cache strategies set for a table or index, and disables or reenables prefetching or MRU strategy.
sp_logiosize	Changes the default I/O size for the log.
sp_spaceused	Provides information about the size of tables and indexes or the amount of space used in a database.
sp_estspace	Estimates the size of tables and indexes, given the number of rows the table will contain.
sp_help	Reports the cache a table is bound to.
sp_helpindex	Reports the cache an index is bound to.
sp_helpdb	Reports the cache a database is bound to.
set showplan on	Reports on I/O size and cache utilization strategies for a query.
set statistics io on	Reports number of reads performed for a query.
set prefetch [on |off]	Enables or disables prefetching for an individual session.
select... (prefetch...lru | mru)	Forces the server to use the specified I/O size or MRU replacement strategy.

Most of the parameters for sp_cacheconfig that configure the data cache are dynamic and do not require that you reboot the server for them to take effect. See Table 19-1 on page 621 for a description of static and dynamic actions.

In addition to using the commands to configure named data caches interactively, you can also edit the configuration file located in the *$SYBASE* directory. However, this requires that you reboot the server. See "Configuring data caches with the configuration file" on page 650 for more information.

Information on data caches

Use sp_cacheconfig to create and configure named data caches. When you first install Adaptive Server, it has a single cache named `default data cache`. To see information about caches, type:

```
sp_cacheconfig
```

The results of sp_cacheconfig look similar to:

```
Cache Name                 Status    Type     Config Value  Run Value
-------------------------  --------- -------- ------------ ------------
default data cache         Active    Default       0.00 Mb    59.44 Mb
                                              ------------ ------------
```

```
                                    Total          0.00 Mb    59.44 Mb
======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 0.00 Mb,   Run Size: 59.44 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ----------- ----------- -----------
    2 Kb   12174 Kb     0.00 Mb     59.44 Mb      10
```

Summary information for each cache is printed in a block at the top of the report, ending with a total size for all configured caches. For each cache, there is a block of information reporting the configuration for the memory pools in the cache.

The columns are:

- Cache Name – gives the name of the cache.
- Status – indicates whether the cache is active. Possible values are:
    - "Pend/Act" – the cache was just created and will be active after a restart.
    - "Active" – the cache is currently active.
    - "Pend/Del" – the cache is active, but will be deleted at the next restart of the server. The cache size was reset to 0 interactively. See "Configuring data caches" on page 620 for more information.
- Type – indicates whether the cache can store data and log pages ("Mixed") or log pages only ("Log Only"). Only the default cache has the type "Default." You cannot change the type of the default data cache or change the type of any other cache to "Default."
- Config Value – displays the currently configured value. In the preceding example output, the default data cache has not been explicitly configured, so its size is 0.
- Run Value – displays the size that Adaptive Server is currently using. For the default data cache, this size is always the amount of all data cache space that is not explicitly configured to another cache.

The second block of output begins with three lines of information that describe the cache. The first two lines repeat information from the summary block at the top. On the third line, “Config Replacement” and “Run Replacement” show the cache policy, which is either “strict LRU” or “relaxed LRU.” The run setting is the setting in effect; if the policy has been changed since the server was restarted, the Config setting will be different from the Run setting.

sp_cacheconfig then provides a row of information for each pool in the cache:

- IO Size – shows the size of the buffers in the pool. The default size of the pool is the size of the server’s logical page. When you first configure a cache, all the space is assigned to the pool. Valid sizes are 2K, 4K, 8K, and 16K.
- Wash Size – indicates the wash size for the pool. See “Changing the wash area for a memory pool” on page 640 for more information.
- Config Size and Run Size – display the configured size and the size currently in use. These values differ for the pool because you cannot explicitly configure its size. These may differ for other pools if you have tried to move space between them, and some of the space could not be freed.
- Config Partition and Run Partition – display the configured number of cache partitions and the number of partitions currently in use. These may differ if you have changed the number of partitions since last restart.
- APF Percent – displays the percentage of the pool that can hold unused buffers brought in by asynchronous prefetch.

A summary line prints the total size of the cache or caches displayed.

Configuring data caches

The default data cache and the procedure cache for Adaptive Server are specified using an absolute value. The first step in planning cache configuration and implementing caches is to set the max memory configuration parameter. After you set max memory, determine how much space you want to allocate for data caches on your server. The size of a data cache is limited only by access to memory on the system; however, max memory should be larger than total logical memory. You must specify an absolute value for the size of the default data cache and all other user-defined caches. For an overview of Adaptive Server memory usage, see Chapter 18, “Configuring Memory.”

You can configure data caches in two ways:

- Interactively, using sp_cacheconfig and sp_poolconfig. This method is dynamic and does not require a reboot of Adaptive Server.
- By editing your configuration file. This method is static and requires that you reboot Adaptive Server.

Each time you change the data cache or execute either sp_cacheconfig or sp_poolconfig, Adaptive Server writes the new cache or pool information into the configuration file and copies the old version of the file to a backup file. A message giving the backup file name is sent to the error log.

Some of the actions you perform with sp_cacheconfig are dynamic and some are static.

Table 19-1: Dynamic and static sp_cacheconfig actions

Dynamic sp_cacheconfig actions	Static sp_cacheconfig actions
Adding a new cache	Changing the number of cache partitions
Adding memory to an existing cache	Reducing a cache size
Deleting a cache	Changing replacement policy
Changing a cache type	

You can reset static parameters by deleting and re-creating the cache:

1. Unbind the cache.
2. Delete the cache.
3. Re-create the cache using the new configuration.
4. Bind objects to the cache.

The following sections describe how to use sp_cacheconfig and sp_poolconfig. See “Configuring data caches with the configuration file” on page 650 for information about using the configuration file.

Mixing static and dynamic parameters

You can specify both static and dynamic parameters in a single command. Adaptive Server performs the dynamic changes immediately and writes all changes to the configuration file (both static and dynamic). Static changes take effect the next time the server is started.

Creating a new cache

The syntax for sp_cacheconfig is:

```
sp_cacheconfig [cachename [ ,"cache_size[P|K|M|G]" ]
      [,logonly | mixed ] [,strict | relaxed ] ]
      [, "cache_partition=[1|2|4|8|16|32|64]"]
```

Size units can be specified with:

- P – pages (Adaptive Server logical page size)
- K – kilobytes (default)
- M – megabytes
- G – gigabytes

See the *Adaptive Server Reference Manual* for a full description of the sp_cacheconfig syntax.

Maximum data cache size is limited only by the amount of memory available on your system. The memory required to create the new cache is taken from Adaptive Server's global memory. When the cache is created:

- It has a default wash size.
- The asynchronous prefetch size is set to the value of global async prefetch limit.
- It has only the default buffer pool.

Use sp_poolconfig to reset these values.

To create a 10MB cache named pubs_cache:

```
sp_cacheconfig pubs_cache, "10M"
```

This command makes changes in the system tables and writes the new values to the configuration file. The cache is immediately active. Running sp_cacheconfig now yields:

```
sp_cacheconfig

Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
default data cache             Active    Default       0.00 Mb      8.00 Mb
pubs_cache                     Active    Mixed        10.00 Mb     10.00 Mb
------------ ------------
Total      10.00 Mb     18.00 Mb
==========================================================================
Cache: default data cache,   Status: Active,   Type: Default
Config Size: 0.00 Mb,   Run Size: 8.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   1636 Kb      0.00 Mb      8.00 Mb     10
==========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
Config Size: 10.00 Mb,   Run Size: 10.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

The pubs_cache is now active, and all space is assigned to the smallest pool.

Note When you create a new cache, the additional memory you specify is validated against max memory. If the sum of total logical memory and additional memory requested is greater than max memory, then Adaptive Server issues an error and does not perform the changes.

You can create as many caches as you want without restarting Adaptive Server.

Insufficient space for new cache

If Adaptive Server cannot allocate all the memory requested, it allocates all the available memory and issues the following message:

```
ASE is unable to get all the memory requested (%d). (%d) kilobytes have been
allocated dynamically.
```

However, this additional memory is not allocated until the next reboot of Adaptive Server.

Adaptive Server notifies you of insufficient space is because some of the memory is unavailable because of resource constraints. System Administrators should make sure these resource constraints are temporary. If the behavior persists, a subsequent reboot may fail.

For example, if max memory is 700MB, pub_cache is 100MB, and the server's total logical memory is 600MB, and you attempt to add 100MB to pub_cache, the additional memory fits into max memory. However, if the server can allocate only 90MB, then it allocates this amount dynamically, but the `size field` of the cache in the configuration file is updated to 100MB. On a subsequent reboot, since Adaptive Server obtains memory for all data caches at one time, the size of pub_cache is 200MB.

Adding memory to an existing named cache

The syntax to add memory to an existing cache is:

sp_cacheconfig *cache_name*, "*new_size*[P|K|M|G]"

The syntax is described in "Creating a new cache" on page 622.

The additional memory you allocate is added to Adaptive Server's page size pool. For example, in a server with a logical page size of 4K, the smallest size for a pool is a 4K buffer pool. If the cache has partitions, the additional memory is divided equally among them.

If there is insufficient memory available, Adaptive Server allocates what it can and then allocates the full amount at the next reboot. See "Insufficient space for new cache" on page 623 for more information.

The following adds 2MB to a cache named pub_cache:

```
sp_cacheconfig pub_cache, "12M"
```

sp_cacheconfig displays the following after the memory is added:

```
                                 sp_cacheconfig pub_cache
 Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
pub_cache                      Active    Mixed      12.00 Mb      12.00
------------ ------------
Total      12.00 Mb     12.00 Mb
==========================================================================
Cache: pub_cache,   Status: Active,   Type: Mixed
Config Size: 12.00 Mb,   Run Size: 12.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2456 Kb      0.00 Mb     12.00 Mb     10
```

This change adds memory to the database page-size buffer pool and recalculates the wash size, as required. If the absolute value is set for wash size, Adaptive Server does not recalculate it.

Decreasing the size of a cache

Note When you reduce the size of a cache, you must reboot Adaptive Server for the change to take effect.

The following is a report on the pubs_log cache:

```
sp_cacheconfig pubs_log

Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_log                 Active    Log Only      7.00 Mb      7.00 Mb
                                            ------------ ------------
                                     Total       7.00 Mb      7.00 Mb
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    920 Kb      0.00 Mb      4.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

The following command reduces the size of the pubs_log cache to 6Mb from command size of 7Mb

```
sp_cacheconfig pubs_log, "6M"
```

After a restart of Adaptive Server, sp_cacheconfig shows:

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_log                 Active    Log Only      6.00 Mb      6.00 Mb
                                            ------------ ------------
                                     Total       6.00 Mb      6.00 Mb
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

When you reduce the size of a data cache, all the space to be removed must be available in the default pool. You may need to move space to the default pool from other pools before you can reduce the size of the data cache. In the last example, if you wanted to reduce the size of the cache to 3MB, you would need to use sp_poolconfig to move some memory into the default pool of 2K from the 4K pool. The memory is moved to "memory available for named caches". See "Changing the size of memory pools" on page 644 for more information.

Deleting a cache

To completely remove a data cache, reset its size to 0:

```
sp_cacheconfig pubs_log, "0"
```

The cache is immediately deleted.

Note You cannot drop the default data cache.

If you delete a data cache, and there are objects bound to the cache, the cache is left in memory and Adaptive Server issues the following message:

```
Cache cache_name not deleted dynamically. Objects are bound to the cache. Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

The entry corresponding to the cache in the configuration file is deleted, as well as the entries corresponding to the cache in sysconfigures, and the cache is deleted the next time Adaptive Server is restarted.

Use sp_helpcache to view all items bound to the cache. Use sp_unbindcache_all to unbind objects. See the *Adaptive Server Reference Manual: Procedures* for more information

If you re-create the cache and restart Adaptive Server, the bindings are marked valid again.

Explicitly configuring the default cache

You must explicitly configure the size of the default data cache because it is specified with an absolute value. Use sp_helpcache to see the amount of memory remaining that can be used for the cache. For example:

```
sp_helpcache
Cache Name                 Config Size    Run Size        Overhead
-----------------------    -------------  ----------      ----------
default data cache         50.00 Mb       50.00 Mb        3.65 Mb
pubs_cache                 10.00 Mb       10.00 Mb        0.77 Mb

Memory Available For       Memory Configured
Named Caches               To Named Caches
--------------------       ----------------
91.26 Mb                   91.25 Mb

------------------ Cache Binding Information: ------------------

Cache Name       Entity Name    Type         Index Name     Status
----------         -----------      -----          ---------        ------
```

To specify the absolute size of the default data cache, execute sp_cacheconfig with default data cache and a size value. This command sets the default data cache size to 25MB:

```
sp_cacheconfig "default data cache", "25M"
```

When you rerun sp_helpconfig, “Config Value” shows the value.

```
sp_cacheconfig

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     50.00 Mb
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                             ------------ ------------
                                      Total      10.00 Mb     60.00 Mb
======================================================================
Cache: default data cache,   Status: Active,   Type: Default
    Config Size: 25.00 Mb,   Run Size: 50.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb 10110 Kb     00.00 Mb    50.00 Mb      10
======================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
    Config Size: 10.00 Mb,   Run Size: 10.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb    0.00 Mb    10.00 Mb    10
```

You can change the size of any named cache by using sp_cacheconfig. Any changes you make to the size of any data cache do not affect the size of any other cache. Similarly, once you specify the size of the default data cache, the configuration of other user-defined caches does not alter the size of the default data cache.

Note If you configure the default data cache and then reduce max memory to a level that sets the total logical memory value higher than the max memory value, Adaptive Server will not start. Edit your configuration file to increase the size of other caches and increase the values of configuration parameters that require memory to create an environment in which total logical memory is higher than max memory. See Chapter 18, "Configuring Memory."for more information.

The default data cache and all user-defined caches are explicitly configured with an absolute value. In addition, many configuration parameters use memory. To maximize performance and avoid errors, set the value of max memory to a level high enough to accommodate all caches and all configuration parameters that use memory.

Adaptive Server issues a warning message if you set max memory to a value less than total logical memory.

Changing the cache type

To reserve a cache for use by only the transaction log, change the cache's type to "logonly." The change is dynamic.

This example creates the cache pubs_log with the type "logonly:"

```
sp_cacheconfig pubs_log, "7M", "logonly"
```

This shows the initial state of the cache:

```
Cache Name          Status    Type     Config Value Run Value
------------------- --------- -------- ------------ ------------
pubs_log            Pend/Act  Log Only      7.00 Mb      0.00 Mb
                                       ------------ ------------
                                Total              7.00 Mb      0.00 Mb
```

You can change the type of an existing "mixed" cache, as long as no non-log objects are bound to it:

```
sp_cacheconfig pubtune_cache, logonly
```

In high-transaction environments, Adaptive Server usually performs best if the default value is twice the server's logical page size (for a server with 2K logical page size, it is 4K, for a server with 4K logical page size, it is 8K, and so on).. For larger page sizes (4, 8, and 16K) use sp_sysmon to find the optimum configuration for your site. For information on configuring caches for improved log performance, see "Matching log I/O size for log caches" on page 634.

Configuring cache replacement policy

If a cache is dedicated to a table or an index, and the cache has little or no buffer replacement when the system reaches a stable state, you can set relaxed LRU (least recently used) replacement policy. Relaxed LRU replacement policy can improve performance for caches where there is little or no buffer replacement occurring, and for most log caches. See "Data cache" in the *Performance and Tuning Guide: Basics* for more information.

To set relaxed replacement policy, use:

```
sp_cacheconfig pubs_log, relaxed
```

The default value is “strict.”

Note Setting the cache replacement policy is not dynamic and requires a restart of Adaptive Server to take effect.

You can create a cache and specify its cache type and the replacement policy in one command. These examples create two caches, pubs_log and pubs_cache:

```
sp_cacheconfig pubs_log, "3M", logonly, relaxed
```

```
sp_cacheconfig pubs_cache, "10M", mixed, strict
```

The change takes place immediately and is made to every cache partition of the named cache.

Here are the results:

```
sp_cacheconfig

Cache Name                      Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     42.29 Mb
pubs_cache               Active    Mixed        10.00 Mb     10.00 Mb
pubs_log                 Active    Log Only      7.00 Mb      7.00 Mb
                                            ------------ ------------
                                     Total      42.00 Mb     59.29 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
=======================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

```
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   1432 Kb      0.00 Mb      7.00 Mb          10
```

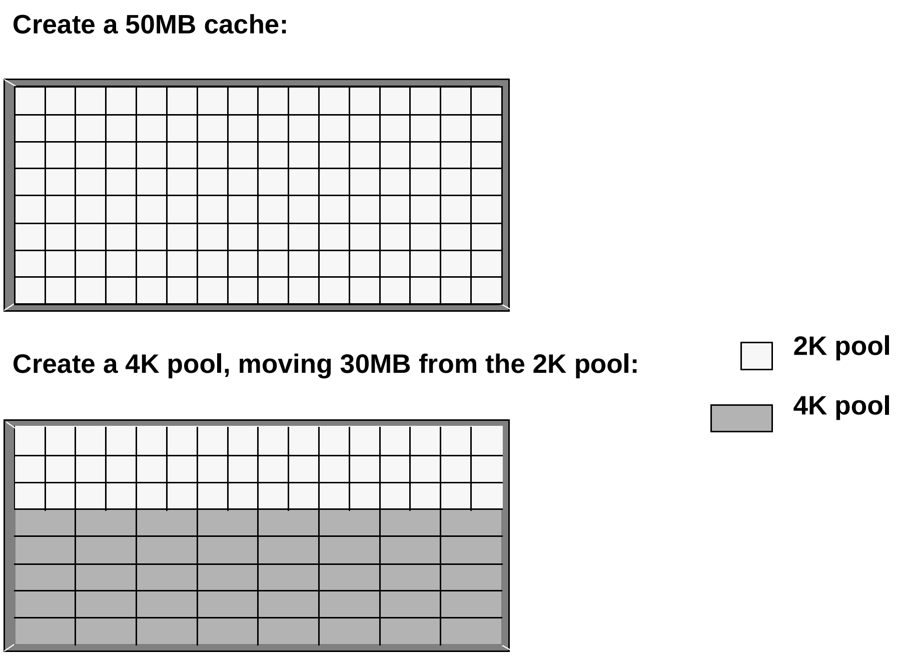
Dividing a data cache into memory pools

After you create a data cache, you can divide it into memory pools, each with a different I/O size. In any cache, you can have only one pool of each I/O size. The minimum size of a memory pool is the size of the server's logical page. Memory pools larger than this must be a power of two and can be a maximum size of one extent.

When Adaptive Server performs large I/Os, multiple pages are read into the cache at the same time. These pages are always treated as a unit; they age in the cache and are written to disk as a unit.

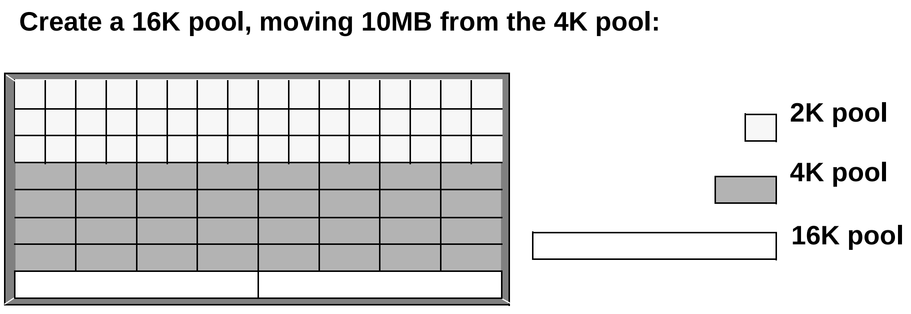
By default, when you create a named data cache, all of its space is assigned to the default memory pool. Creating additional pools reassigns some of that space to other pools, reducing the size of the default memory pool. For example, if you create a data cache with 50MB of space, all the space is assigned to the 2K pool. If you configure a 4K pool with 30MB of space in this cache, the 2K pool is reduced to 20MB.

Figure 19-2: Configuring a cache and a 4K memory pool



After you create the pools, you can move space between them. For example, in a cache with a 20MB 2K pool and a 30MB 4K pool, you can configure a 16K pool, taking 10MB of space from the 4K pool.

Figure 19-3: Moving space from an existing pool to a new pool



The commands that move space between pools within a cache do not require a restart of Adaptive Server, so you can reconfigure pools to meet changing application loads with little impact on server activity.

In addition to creating pools in the caches you configure, you can add memory pools for I/Os up to 16K to the default data cache.

The syntax for configuring memory pools is:

```
sp_poolconfig cache_name, "memsize[P|K|M|G]",
"config_poolK" [, "affected_poolK"]
```

Pool configuration sets the config_pool to the size specified in the command. It always affects a second pool (the affected_pool) by moving space to or from that pool. If you do not specify the affected_pool, the space is taken from or allocated to the 2K pool. The minimum size for a pool is 512K.

This example creates a 7MB pool of 16K pages in the pubs_cache data cache:

```
sp_poolconfig pubs_cache, "7M", "16K"
```

This command reduces the size of the memory pool. To see the current configuration, run sp_cacheconfig, giving only the cache name:

```
sp_cacheconfig pubs_cache

Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_cache               Active    Mixed       10.00 Mb     10.00 Mb
                                            ------------ ------------
                                     Total     10.00 Mb     10.00 Mb
=======================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:              1,   Run Partition:              1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb      3.00 Mb     10


   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

You can also create memory pools in the default data cache.

In the following example, you start with this cache configuration:

```
Cache Name                 Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default     25.00 Mb     42.29 Mb
                                                    ------------ ------------
                                           Total      25.00 Mb     42.29 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
```

```
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
```

This command creates a 16K pool in the default data cache that is 8 megabytes:

```
sp_poolconfig "default data cache", "8M", "16K"
```

It results in this configuration, reducing the “Run Size” of the 2K pool:

```
Cache Name                 Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     42.29 Mb
                                             ------------ ------------
                                       Total      25.00 Mb     42.29 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     34.29 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

You do not need to configure the size of the 2K memory pool in caches that you create. Its “Run Size” represents all the memory not explicitly configured to other pools in the cache.

Matching log I/O size for log caches

If you create a cache for the transaction log of a database, configure most of the space in that cache to match the log I/O size. The default value is twice the server’s logical page size (for a server with 2K logical page size, it is 4K, for a server with 4K logical page size, it is 8K, and so on). Adaptive Server uses 2K I/O for the log if a 4K pool is not available. The log I/O size can be changed with sp_logiosize. The log I/O size of each database is reported in the error log when Adaptive Server starts, or you can check the size of a database by using the database and issuing sp_logiosize with no parameters.

This example creates a 4K pool in the pubs_log cache:

```
sp_poolconfig pubs_log, "3M", "4K"
```

You can also create a 4K memory pool in the default data cache for use by transaction logs of any databases that are not bound to another cache:

```
sp_poolconfig "default data cache", "2.5M", "4K"
```

See “Choosing the I/O size for the transaction log” in the *Performance and Tuning Guide: Basics* for information on tuning the log I/O size.

Binding objects to caches

sp_bindcache assigns a database, table, index or text/image object to a cache. Before you can bind an entity to a cache, the following conditions must be met:

- The named cache must exist, and its status must be “Active.”
- The database or database object must exist.
- To bind tables, indexes, or objects, you must be using the database where they are stored.
- To bind system tables, including the transaction log table syslogs, the database must be in single-user mode.
- To bind a database, you must be using the master database.
- To bind a database, user table, index, text object, or image object to a cache, the type of cache must be “Mixed.” Only the syslogs table can be bound to a cache of “Log Only” type.
- You must own the object or be the Database Owner or the System Administrator.

Binding objects to caches is dynamic.

The syntax for binding objects to caches is:

```
sp_bindcache cache_name, dbname [,[owner.]tablename
[, indexname |  "text only" ] ]
```

The owner name is optional if the table is owned by “dbo.”

This command binds the titles table to the pubs_cache:

```
sp_bindcache  pubs_cache, pubs2, titles
```

To bind an index on titles, add the index name as the third parameter:

```
sp_bindcache pubs_cache, pubs2,  titles, titleind
```

The owner name is not needed in the examples above because the objects in the pubs2 database are owned by "dbo." To specify a table owned by any other user, add the owner name. You must enclose the parameter in quotation marks, since the period in the parameter is a special character:

```
sp_bindcache pubs_cache, pubs2, "fred.sales_east"
```

This command binds the transaction log, syslogs, to the pubs_log cache:

```
sp_bindcache pubs_log, pubs2, syslogs
```

The database must be in single-user mode before you can bind any system tables, including the transaction log, syslogs, to a cache. Use sp_dboption from master, and a use database command, and run checkpoint:

```
sp_dboption pubs2, single, true
use pubs2
checkpoint
```

text and image columns for a table are stored in a separate data structure in the database. To bind this object to a cache, add the "text-only" parameter:

```
sp_bindcache pubs_cache, pubs2, au_pix, "text only"
```

This command, executed from master, binds the tempdb database to a cache:

```
sp_bindcache tempdb_cache, tempdb
```

You can rebind objects without dropping existing bindings.

Cache binding restrictions

You cannot bind or unbind a database object when:

- Dirty reads are active on the object.
- A cursor is open on the object

In addition, Adaptive Server needs to lock the object while the binding or unbinding takes place, so the procedure may have a slow response time, because it waits for locks to be released. See "Locking to perform bindings" on page 650 for more information.

Getting information about cache bindings

sp_helpcache provides information about a cache and the entities bound to it when you provide the cache name:

```
sp_helpcache pubs_cache

Cache Name           Config Size     Run Size       Overhead
-------------------- ------------    ----------     ----------
pubs_cache             10.00 Mb      10.00 Mb        0.77 Mb

------------------ Cache Binding Information: ------------------

Cache Name      Entity Name           Type   Index Name  Status
----------      -----------           ----   ----------  ------
pubs_cache      pubs2.dbo.titles      index  titleind    V
pubs_cache      pubs2.dbo.au_pix      index  tau_pix     V
pubs_cache      pubs2.dbo.titles      table              V
pubs_cache      pubs2.fred.sales_east table              V
```

If you use sp_helpcache without a cache name, it prints information about all the configured caches on Adaptive Server and all the objects that are bound to them.

sp_helpcache performs string matching on the cache name, using *%cachename%*. For example, “pubs” matches both “pubs_cache” and “pubs_log”.

The “Status” column reports whether a cache binding is valid (“V”) or invalid (“I”). If a database or object is bound to a cache, and the cache is deleted, binding information is retained in the system tables, but the cache binding is marked as invalid. All objects with invalid bindings use the default data cache. If you subsequently create another cache with the same name, the binding becomes valid when the cache is activated.

Checking cache overhead

Note The cache overhead accounting for Adaptive Server versions 12.5.1 and later is more explicit than earlier versions of Adaptive Server. The actual overhead is the same, but instead of being part of the server overhead, it is now considered as part of the cache overhead.

sp_helpcache can report the amount of overhead required to manage a named data cache of a given size. When you create a named data cache, all the space you request with sp_cacheconfig is made available for cache space. The memory needed for cache management is taken from the default data cache.

To see the overhead required for a cache, give the proposed size. You can use P for pages, K for kilobytes, M for megabytes, or G for gigabytes. The following example checks the overhead for 20,000 pages:

```
sp_helpcache "20000P"

2.96Mb of overhead memory will be needed to manage a
cache of size 20000P
```

You are not wasting any cache space by configuring user caches. About 5 percent of memory is required for the structures that store and track pages in memory, whether you use a single large data cache or several smaller caches.

How overhead affects total cache space

The example detailed in “Information on data caches” on page 618 shows a default data cache with 59.44 MB of cache space available before any user-defined caches are created. The server in this example uses a 2K logical page. When the 10MB pubs_cache is created, the results of sp_cacheconfig show a total cache size of 59.44 MB.

Configuring a data cache can appear to increase or decrease the total available cache. The explanation for this lies in the amount of overhead required to manage a cache of a particular size, and the fact that the overhead is not included in the values displayed by sp_cacheconfig.

Using sp_helpcache to check the overhead of the original 59.44MB default cache and the new 10MB cache shows that the change in space is due to changes in the size of overhead. The following command shows the overhead for the default data cache before any changes were made:

```
sp_helpcache "59.44M"

4.24Mb of overhead memory will be needed to manage a
cache of size 59.44M
```

This command shows the overhead for pubs_cache:

```
sp_helpcache "10M"

0.77Mb of overhead memory will be needed to manage a
cache of size 10M
```

The following calculations add the overhead required to manage the original cache space and then subtract the overhead for pubs_cache:

Original total cache size (overhead not included)	**59.44**
Overhead for 59.44 MB default cache	+4.24
Total cache space, including overhead	63.68
10MB pubs_cache and .53MB overhead	-10.77
Remaining space	52.91
Overhead for 52.95MB cache	- 3.83
Usable size of the default cache	49.08

Cache sizes are rounded to two places when printed by sp_cacheconfig, and overhead is rounded to two places by sp_helpcache, so you will see a small amount of rounding error in the output.

Dropping cache bindings

Two commands drop cache bindings:

- sp_unbindcache unbinds a single entity from a cache.
- sp_unbindcache_all unbinds all objects bound to a cache.

The syntax for sp_unbindcache is:

```
sp_unbindcache dbname [,[owner.]tablename
[, indexname | "text only"] ]
```

This commands unbinds the titleidind index on the titles table in the pubs2 database:

```
sp_unbindcache pubs2, titles, titleidind
```

To unbind all the objects bound to a cache, use sp_unbindcache_all, giving the cache’s name:

```
sp_unbindcache_all pubs_cache
```

Note You cannot use sp_unbindcache_all if more than eight databases and/or objects in eight databases are bound to the cache. You must use sp_unbindcache on individual databases or objects to reduce the number of databases involved to eight or less.

When you drop a cache binding for an object, all the pages currently in memory are cleared from the cache.

Changing the wash area for a memory pool

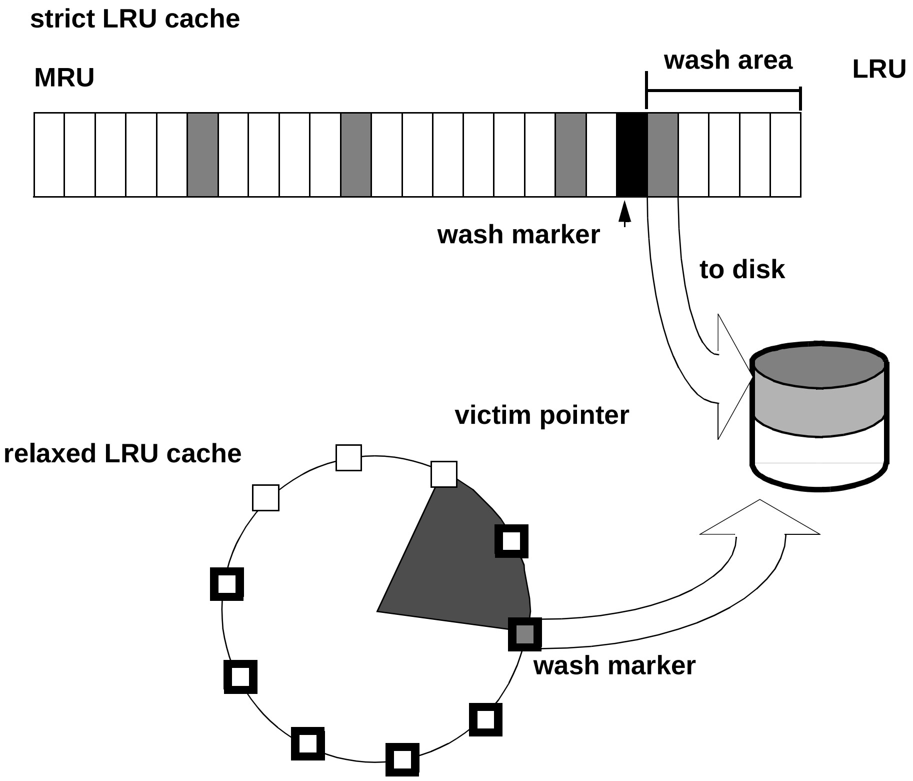
When Adaptive Server needs to read a buffer into cache, it places the buffer:

- At the LRU (least recently used) end of each memory pool, in a cache with strict LRU policy
- At the victim pointer, in a cache with relaxed LRU policy. If the recently used bit of buffer at the victim marker is set, the victim pointer is moved to the next buffer in the pool.

A portion of each pool is configured as the **wash area**. After dirty pages (pages that have been changed in cache) pass the wash marker and enter the wash area, Adaptive Server starts an asynchronous I/O on the page. When the write completes, the page is marked clean and remains available in the cache.

The space in the wash area must be large enough so that the I/O on the buffer can complete before the page needs to be replaced. Figure 19-4 illustrates how the wash area of a buffer pool works with a strict and relaxed LRU cache.

Figure 19-4: Wash area of a buffer pool



By default, the size of the wash area for a memory pool is configured as follows:

- If the pool size is less than 300MB, the default wash size is 20 percent of the buffers in the pool.
- If the pool size is greater than 300MB, the default wash size is 20 percent of the number of buffers in 300MB.

The minimum wash size is 10 buffers. The maximum size of the wash area is 80 percent of the pool size.

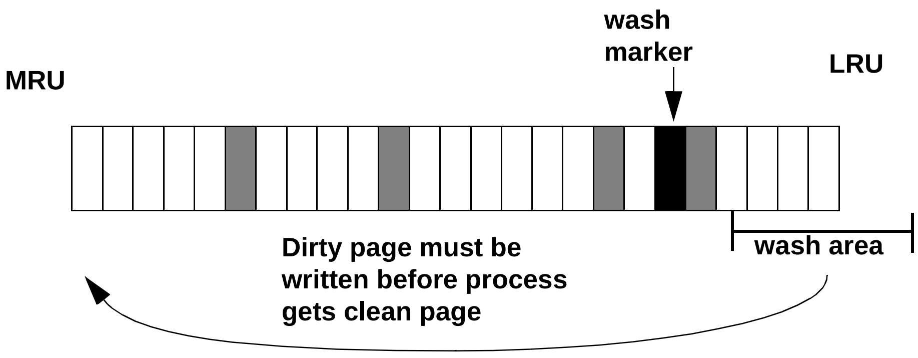
A buffer is a block of pages that matches the I/O size for the pool. Each buffer is treated as a unit: all pages in the buffer are read into cache, written to disk, and aged in the cache as a unit. For the size of the block, multiply the number of buffers by the pool size—for a 2K pool, 256 buffers equals 512K; for a 16K pool, 256 buffers equals 4096K.

For example, if you configure a 16K pool with 1MB of space, the pool has 64 buffers; 20 percent of 64 is 12.8. This is rounded down to 12 buffers, or 192K, are allocated to the wash area.

When the wash area is too small

If the wash area is too small for the usage in a buffer pool, operations that need a clean buffer may have to wait for I/O to complete on the dirty buffer at the LRU end of the pool or at the victim marker. This is called a **dirty buffer grab**, and it can seriously impact performance. Figure 19-5 shows a dirty buffer grab on a strict replacement policy cache.

Figure 19-5: Small wash area results in a dirty buffer grab



You can use sp_sysmon to determine whether dirty buffer grabs are taking place in your memory pools. Run sp_sysmon while the cache is experiencing a heavy period of I/O and heavy update activity, since it is the combination of many dirty pages and high cache replacement rates that usually causes dirty buffer grabs.

If the “Buffers Grabbed Dirty” output in the cache summary section shows a nonzero value in the “Count” column, check the “Grabbed Dirty” row for each pool to determine where the problem lies. Increase the size of the wash area for the affected pool. This command sets the wash area of the 8K memory pool to 720K:

```
sp_poolconfig pubs_cache, "8K", "wash=720K"
```

If the pool is very small, you may also want to increase its pool size, especially if sp_sysmon output shows that the pool is experiencing high turnover rates.

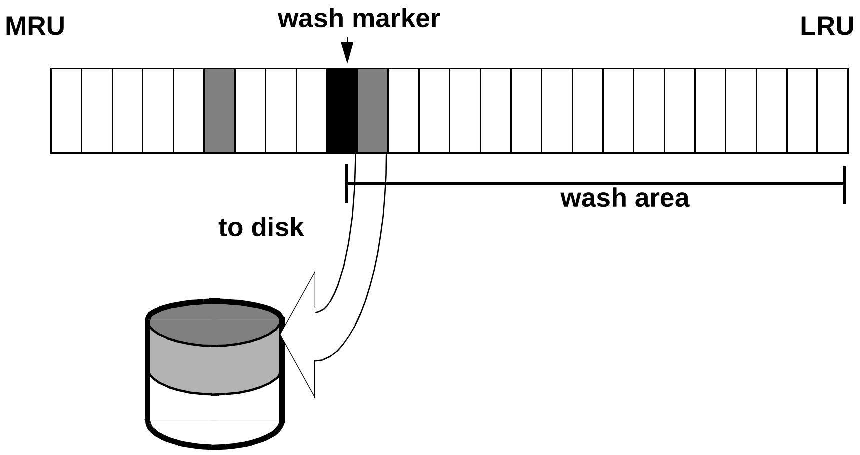
See the *Performance and Tuning Guide* for more information.

When the wash area is too large

If the wash area is too large in a pool, the buffers move too quickly past the "wash marker" in cache, and an asynchronous write is started on any dirty buffers, as shown in Figure 19-6. The buffer is marked "clean" and remains in the wash area of the MRU/LRU chain until it reaches the LRU. If another query changes a page in the buffer, Adaptive Server must perform additional I/O to write it to disk again.

If sp_sysmon output shows a high percentage of buffers "Found in Wash" for a strict replacement policy cache, and there are no problems with dirty buffer grabs, you may want to try reducing the size of the wash area. See the *Performance and Tuning Guide* for more information.

Figure 19-6: Effects of making the wash area too large



Changing the asynchronous prefetch limit for a pool

The asynchronous prefetch limit specifies the percentage of the pool that can be used to hold pages that have been brought into the cache by asynchronous prefetch, but have not yet been used by any queries. The default value for the server is set with the global async prefetch limit configuration parameter. Pool limits, set with sp_poolconfig, override the default limit for a single pool.

This command sets the percentage for the 2K pool in the pubs_cache to 20:

```
sp_poolconfig pubs_cache, "2K", "local async prefetch limit=20"
```

Changes to the prefetch limit for a pool take effect immediately and do not require a restart of Adaptive Server. Valid values are 0–100. Setting the prefetch limit to 0 disables asynchronous prefetching in a pool. For information about the impact of asynchronous prefetch on performance, see See Chapter 10, “Tuning Asynchronous Prefetch,” in *Performance and Tuning Guide: Optimizer and Abstract Plans*.

Changing the size of memory pools

To change the size of a memory pool, use sp_poolconfig to specify the cache, the new size for the pool, the I/O size of the pool you want to change, and the I/O size of the pool from which the buffers should be taken. If you do not specify the final parameter, all the space is taken from or assigned to the pool.

Moving space from the memory pool

This command checks the current configuration of the pubs_log cache (The output in this example is based on the examples in the previous sections):

```
sp_cacheconfig pubs_log

Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_log                 Active    Log Only      6.00 Mb      6.00 Mb
                                            ------------ ------------
                                     Total       6.00 Mb      6.00 Mb
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

This command increases the size of the 4K pool to 5MB, moving the required space from the 2K pool:

```
sp_poolconfig pubs_log, "5M", "4K"
```

```
sp_cacheconfig pubs_log
```

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
pubs_log                 Active    Log Only      6.00 Mb      6.00 Mb
                                            ------------ ------------
                                     Total      6.00 Mb      6.00 Mb
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:           1,   Run Partition:           1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
```

Moving space from other memory pools

To transfer space from another pool specify the cache name, a “to” I/O size, and a “from” I/O size. This output shows the current configuration of the default data cache:

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
                                            ------------ ------------
                                     Total     25.00 Mb     29.28 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:           1,   Run Partition:           1
```

```
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

The following command increases the size of the 4K pool from 2.5MB to 4MB, taking the space from the 16K pool:

```
sp_poolconfig "default data cache","4M", "4K","16K"
```

This results in the following configuration:

```
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
                                            ------------ ------------
                                      Total     25.00 Mb     29.28 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
   16 Kb   1632 Kb      6.50 Mb      6.50 Mb     10
```

When you issue a command to move buffers between pools in a cache, Adaptive Server can move only “free” buffers. It cannot move buffers that are in use or buffers that contain changes that have not been written to disk.

When Adaptive Server cannot move as many buffers as you request, it displays an informational message, giving the requested size and the resulting size of the memory pool.

Adding cache partitions

On multi-engine servers, more than one task can attempt to access the cache at the same time. By default, each cache has a single spinlock, so that only one task can change or access the cache at a time. If cache spinlock contention is above 10 percent, increasing the number of cache partitions for a cache can reduce spinlock contention, and improve performance.

You can configure the number of cache partitions for:

- All data caches, using the global cache partition number configuration parameter
- An individual cache, using sp_cacheconfig

The number of partitions in a cache is always a power of 2 between 1 and 64. No pool in any cache partition can be smaller than 512K. In most cases, since caches can be sized to meet requirements for storing individual objects, you should use the local setting for the particular cache where spinlock contention is an issue.

See "Reducing spinlock contention with cache partitions" in the *Performance and Tuning Guide: Basics* for information on choosing the number of partitions for a cache.

Setting the number of cache partitions with *sp_configure*

Use sp_configure to set the number of cache partitions for all caches on a server. For example, to set the number of cache partitions to 2, enter:

```
sp_configure "global cache partition number",2
```

You must restart the server for the change to take effect.

Setting the number of local cache partitions

Use sp_cacheconfig or the configuration file to set the number of local cache partitions. This command sets the number of cache partitions in the default data cache to 4:

```
sp_cacheconfig "default data cache", "cache_partition=4"
```

You must restart the server for the change to take effect.

Precedence

The local cache partition setting always takes precedence over the global cache partition value.

These commands set the server-wide partition number to 4, and the number of partitions for pubs_cache to 2:

```
sp_configure "global cache partition number", 4
sp_cacheconfig "pubs_cache", "cache_partition=2"
```

The local cache partition number takes precedence over the global cache partition number, so pubs_cache uses 2 partitions. All other configured caches have 4 partitions.

To remove the local setting for pubs_cache, and use the global value instead, use this command:

```
sp_cacheconfig "pubs_cache", "cache_partition=default"
```

To reset the global cache partition number to the default, use:

```
sp_configure "global cache partition number", 0, "default"
```

Dropping a memory pool

To completely remove a pool, reset its size to 0. The following removes the 16K pool and places all space in the default pool:

```
sp_poolconfig "default data cache", "0",  "
16K"sp_cacheconfig "default data cache"
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
                                            ------------ ------------
                                     Total      25.00 Mb     29.28 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
```

If you do not specify the affected pool size (16K in the example above), all the space is placed in the default pool. You cannot delete the default pool in any cache.

When pools cannot be dropped due to pages use

If the pool you are trying to delete contains pages that are in use, or pages that have dirty reads, but are not written to disk, Adaptive Server moves as many pages as possible to the specified pool and prints an informational message telling you the size of the remaining pool. If the pool size is smaller than the minimum allowable pool size, you also receive a warning message saying the pool has been marked unavailable. If you run sp_cacheconfig after receiving one of these warnings, the pool detail section for these pools contains an extra "Status" column, with either "Unavailable/too small" or "Unavailable/deleted" for the affected pool.

You can reissue the command at a later time to complete removing the pool. Pools with "Unavailable/too small" or "Unavailable/deleted" are also removed when you restart Adaptive Server.

Cache binding effects on memory and query plans

Binding and unbinding objects may have an impact on performance. When you bind or unbind a table or an index:

- The object's pages are flushed from the cache.
- The object must be locked to perform the binding.
- All query plans for procedures and triggers must be recompiled.

Flushing pages from cache

When you bind an object or database to a cache, the object's pages that are already in memory are removed from the source cache. The next time the pages are needed by a query, they are read into the new cache. Similarly, when you unbind objects, the pages in cache are removed from the user-configured cache and read into the default cache the next time they are needed by a query.

Locking to perform bindings

To bind or unbind user tables, indexes, or text or image objects, the cache binding commands need an exclusive table lock on the object. If a user holds locks on a table, and you issue sp_bindcache, sp_unbindcache, or sp_unbindcache_all on the object, the system procedure sleeps until it can acquire the locks it needs.

For databases, system tables, and indexes on system tables, the database must be in single-user mode, so there cannot be another user who holds a lock on the object.

Cache binding effects on stored procedures and triggers

Cache bindings and I/O sizes are part of the query plan for stored procedures and triggers. When you change the cache binding for an object, all the stored procedures that reference the object are recompiled the next time they are executed. When you change the cache binding for a database, all stored procedures that reference any objects in the database that are not explicitly bound to a cache are recompiled the next time they are run.

Configuring data caches with the configuration file

You can add or drop named data caches and reconfigure existing caches and their memory pools by editing the configuration file that is used when you start Adaptive Server.

Note You cannot reconfigure caches and pools on a server while it is running. Any attempt to read a configuration file that contains cache and pool configurations different from those already configured on the server causes the read to fail.

Cache and pool entries in the configuration file

Each configured data cache on the server has this block of information in the configuration file:

```
[Named Cache:cache_name]
     cache size = {size | DEFAULT}
     cache status = {mixed cache | log only | default data cache}
     cache replacement policy = {DEFAULT |
          relaxed LRU replacement| strict LRU replacement }
```

Size units can be specified with:

- P – pages (Adaptive Server pages)
- K – kilobytes (default)
- M – megabytes
- G – gigabytes

This example shows the configuration file entry for the default data cache:

```
[Named Cache:default data cache]
      cache size = DEFAULT
      cache status = default data cache
      cache replacement policy = strict LRU replacement
```

The default data cache entry is the only cache entry that is required in for Adaptive Server to start. It must have the cache size and cache status, and the status must be "default data cache."

If the cache has pools configured in addition to the pool, the block in the preceding example is followed by a block of information for each pool:

```
[16K I/O Buffer Pool]
        pool size = size
        wash size = size
        local async prefetch limit = DEFAULT
```

Note In some cases, there is no configuration file entry for the pool in a cache. If you change the asynchronous prefetch percentage with sp_poolconfig, the change is not written to the configuration file, only to system tables.

This example shows output from sp_cacheconfig, followed by the configuration file entries that match this cache and pool configuration:

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      29.28 Mb     25.00 Mb
pubs_cache                Active    Mixed        20.00 Mb     20.00 Mb
pubs_log                  Active    Log Only      6.00 Mb     6.00 Mb
tempdb_cache              Active    Mixed         4.00 Mb     4.00 Mb
                                             ------------ ------------
```

```
                                    Total      59.28 Mb     55.00 Mb
=======================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 29.28 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
=======================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 20.00 Mb,   Run Size: 20.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2662 Kb      0.00 Mb     13.00 Mb     10
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
=======================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
=======================================================================
Cache: tempdb_cache,   Status: Active,   Type: Mixed
     Config Size: 4.00 Mb,   Run Size: 4.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    818 Kb      0.00 Mb      4.00 Mb     10
```

This is the matching configuration file information:

```
[Named Cache:default data cache]
        cache size = 29.28M
        cache status = default data cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT

[2K I/O Buffer Pool]
        pool size = 6656.0000k
        wash size = 3844 K
        local async prefetch limit = DEFAULT

[4K I/O Buffer Pool]
        pool size = 4.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_cache]
        cache size = 20M
        cache status = mixed cache
        cache replacement policy = strict LRU
replacement
        local cache partition number = DEFAULT

[16K I/O Buffer Pool]
        pool size = 7.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_log]
        cache size = 6M
        cache status = log only
        cache replacement policy = relaxed LRU
replacement
        local cache partition number = DEFAULT

[4K I/O Buffer Pool]
        pool size = 5.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT


[Named Cache:tempdb_cache]
        cache size = 4M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT
```

For more information about the configuration file, see Chapter 4, "Setting Configuration Parameters."

Warning! Check the max memory configuration parameter and allow enough memory for other Adaptive Server needs. If you assign too much memory to data caches in your configuration file, Adaptive Server will not start. If this occurs, edit the configuration file to reduce the amount of space in the data caches, or increase the max memory allocated to Adaptive Server. See Chapter 18, "Configuring Memory." for suggestions on monitoring cache sizes.

Cache configuration guidelines

User-definable caches are a performance feature of Adaptive Server. This chapter addresses only the mechanics of configuring caches and pools and binding objects to caches. Performance information and suggested strategies for testing cache utilization is addressed in Chapter 10, "Memory Use and Performance," in the *Performance and Tuning Guide: Basics*.

Here are some general guidelines:

- The size of the default data cache does not decrease when you configure other caches.
- Make sure that your default data cache is large enough for all cache activity on unbound tables and indexes. All objects that are not explicitly bound to a cache use the default cache. This includes any unbound system tables in the user databases, the system tables in master, and any other objects that are not explicitly bound to a cache.
- During recovery, only the 2K memory pool of the default cache is active. Transactions logs are read into the 2K pool of the default cache. All transactions that must be rolled back or rolled forward must read data pages into the default data cache. If the default data cache is too small, it can slow recovery time.
- Do not "starve" the 2K pool in any cache. For many types of data access, there is no need for large I/O. For example, a simple query that uses an index to return a single row to the user might use 4 or 5 2K I/Os, and gain nothing from 16K I/O.
- Certain commands can perform only 2K I/O: disk init, certain dbcc commands, and drop table. dbcc checktable can perform large I/O, and dbcc checkdb performs large I/O on tables and 2K I/O on indexes.

- For caches used by transaction logs, configure an I/O pool that matches the default log I/O size. This size is set for a database using sp_logiosize. The default value is 4K.
- Trying to manage every index and object and its caching can waste cache space. If you have created caches or pools that are not optimally used by the tables or indexes bound to them, they are wasting space and creating additional I/O in other caches.
- If tempdb is used heavily by your applications, bind it to its own cache. Note that you can bind only the entire tempdb database, you cannot bind individual objects from tempdb.
- For caches with high update and replacement rates, be sure that your wash size is large enough.
- On multi-CPU systems, spread your busiest tables and their indexes across multiple caches to avoid spinlock contention.
- Consider reconfiguring caches or the memory pools within caches to match changing workloads. Reconfiguring caches requires a restart of the server, but memory pool reconfiguration does not.

  For example, if your system performs mostly OLTP (online transaction processing) during most of the month, and has heavy DSS (decision-support system) activity for a few days, consider moving space from the 2K pool to the 16K pool for the high DSS activity and resizing the pools for OLTP when the DSS workload ends.

Configuration file errors

If you edit your configuration file manually, check the cache, pool, and wash sizes carefully. Certain configuration file errors can cause start-up failure:

- The total size of all of the caches cannot be greater than the amount of max memory, minus other Adaptive Server memory needs.
- The total size of the pools in any cache cannot be greater than the size of the cache.
- The wash size cannot be too small (less than 20 percent of the pool size, with a minimum of 10 buffers) and cannot be larger than 80 percent of the buffers in the pool.
- The default data cache status must be “default data cache,” and the size must be specified, either as a numeric value or as “DEFAULT”.
- The status and size for any cache must be specified.

- The pool size and wash size for all pools larger than 2K must be specified.
- The status of all user-defined caches must be “mixed cache” or “log only”.
- The cache replacement policy and the asynchronous prefetch percentage are optional, but, if specified, they must have correct parameters or “DEFAULT”.

In most cases, problems with missing entries are reported as “unknown format” errors on lines immediately following the entry where the size, status, or other information was omitted. Other errors provide the name of the cache where the error occurred and the type of error. For example, you see this error if the wash size for a pool is specified incorrectly:

```
The wash size for the 4k buffer pool in cache pubs_cache
has been incorrectly configured. It must be a minimum
of 10 buffers and a maximum of 80 percent of the number
of buffers in the pool.
```

CHAPTER 20

Managing Multiprocessor Servers

This chapter provides guidelines for administering Adaptive Server on a multiprocessor.

Parallel processing

Adaptive Server implements the Sybase Virtual Server Architecture™, which enables it to take advantage of the parallel processing feature of symmetric multiprocessing (SMP) systems. You can run Adaptive Server as a single process or as multiple, cooperating processes, depending on the number of CPUs available and the demands placed on the server machine. This chapter describes:

- The target machine architecture for the SMP Adaptive Server
- Adaptive Server architecture for SMP environments
- Adaptive Server task management in the SMP environment
- Managing multiple engines

For information on application design for SMP systems, see Chapter 20, “Managing Multiprocessor Servers,” in the *Performance and Tuning: Basics*.

Definitions

Here are the definitions for several terms used in this chapter:

- **Process** – an execution environment scheduled onto physical CPUs by the operating system.
- **Engine** – a process running an Adaptive Server that communicates with the other Adaptive Server processes via shared memory. An engine can be thought of as one CPU's worth of processing power. It does *not* represent a particular CPU. Also referred to as a **server engine**.
- **Task** – an execution environment within the Adaptive Server that is scheduled onto engines by the Adaptive Server.
- **Affinity** – describes a process in which a certain Adaptive Server task runs only on a certain engine (*task affinity*), a certain engine handles network I/O for a certain task (*network I/O affinity*), or a certain engine runs only on a certain CPU (*engine affinity*).
- **Network affinity migration** – describes the process of moving network I/O from one engine to another. SMP systems that support this migration allow Adaptive Server to distribute the network I/O load among all of its engines.

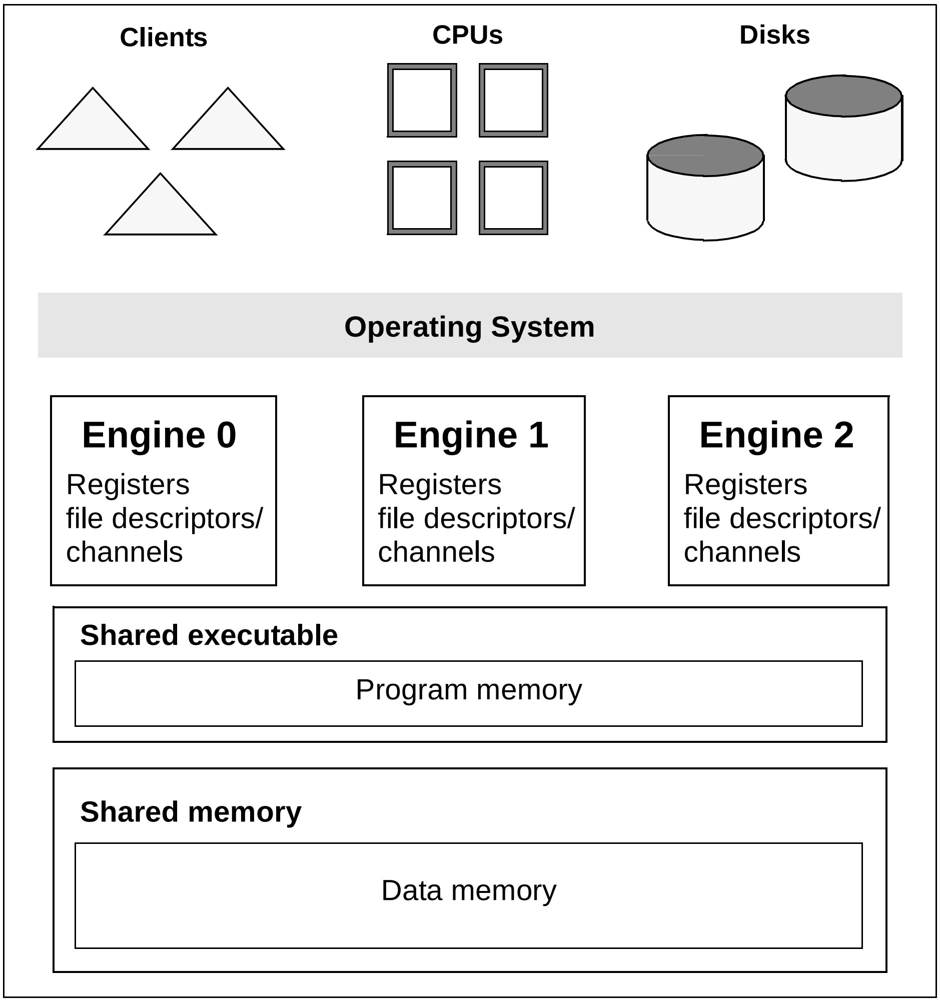
Target architecture

The SMP environment product is intended for machines with the following features:

- A symmetric multiprocessing operating system
- Shared memory over a common bus
- 1–128 processors
- No master processor
- Very high throughput

Adaptive Server consists of one or more cooperating processes (called engines), all of which run the server program in parallel. See Figure 20-1.

Figure 20-1: SMP environment architecture



When clients connect to Adaptive Server, the client connections are assigned to engines in a round-robin fashion, so all engines share the work of handling network I/O for clients. All engines are peers, and they communicate via shared memory.

The server engines perform all database functions, including updates and logging. Adaptive Server, not the operating system, dynamically schedules client tasks onto available engines.

The operating system schedules the engine processes onto physical processors. Any available CPU is used for any engine; there is no *engine affinity*. The processing is called *symmetric* because the lack of affinity between processes and CPUs creates a symmetrically balanced load.

Configuring an SMP environment

Configuring the SMP environment is much the same as configuring the uniprocessor environment, although SMP machines are typically more powerful and handle many more users. The SMP environment provides the additional ability to control the number of engines.

Managing engines

To achieve optimum performance from an SMP system, you must maintain the right number of engines.

An engine represents a certain amount of CPU power. It is a configurable resource like memory.

Note If your server connections use CIS, they are affinitied to a single engine, and will not be allowed to migrate from one engine to another. Adaptive Server uses a load balancing algorithm to evenly distribute the load among the engines.

Resetting the number of engines

When you first install Adaptive Server, the system is configured for a single engine. To use multiple engines, you must reset the number of engines the first time you restart the server. You may also want to reset the number of engines at other times. For example, you might want to:

- Increase the number of engines if current performance is not adequate for an application and there are enough CPUs on the machine.
- Decrease the number of engines if a hardware failure disables CPUs on the machine.

max online engines controls the number of engines used by Adaptive Server. Reset this parameter with sp_configure. For example, to set the number of engines to 3, issue the following:

```
sp_configure "max online engines", 3
```

You must restart the server to reset the number of engines.

Repeat these steps whenever you need to change the number of engines. Engines other than engine 0 are brought online after recovery is complete.

Choosing the right number of engines

It is important that you choose the right number of engines for Adaptive Server. Here are some guidelines:

- *Never* have more engines than CPUs. Doing so may slow performance. If a CPU goes offline, use sp_configure to reduce the max online engines configuration parameter by 1 and restart Adaptive Server.
- Have only as many engines as you have *usable* CPUs. If there is a lot of processing by the client or other non-Adaptive Server processes, then one engine per CPU may be excessive. Remember, too, that the operating system may take up part of one of the CPUs.
- Have *enough* engines. It is good practice to start with a few engines and add engines when the existing CPUs are almost fully used. If there are too few engines, the capacity of the existing engines will be exceeded and bottlenecks may result.

Starting and stopping engines

This section describes how to start and stop Adaptive Server engines using sp_engine.

Monitoring engine status

Before you bring an engine on- or offline, you may need to check the status of the engines currently running. sysengines includes any of the following in the status column:

- online – indicates the engine is online.

- in offline – indicates that sp_engine offline has been run. The engine is still allocated to the server, but is in the process of having its tasks migrated to other engines.
- in destroy – indicates that all tasks have successfully migrated off the engine, and that the server is waiting on the OS level task to deallocate the engine.
- in create – indicates that an engine is in the process of being brought online.

The following command shows the engine number, status, number of tasks affinitied, and the time an engine was brought online:

```
select engine, status, affinitied, starttime
from sysengines
```

```
engine status       affinitied  starttime
------ ------------ ----------- --------------------------
     0 online                12        Mar  5 2001 9:40PM
     1 online                 9        Mar  5 2001 9:41PM
     2 online                12        Mar  5 2001 9:41PM
     3 online                14        Mar  5 2001 9:51PM
     4 online                 8        Mar  5 2001 9:51PM
     5 in offline            10        Mar  5 2001 9:51PM
```

Starting and stopping engines with *sp_engine*

You can dynamically stop or start engines using sp_engine, which allows a System Administrator to reconfigure CPU resources as processing requirements fluctuate over time.

The syntax for sp_engine is:

sp_engine {“online” | [offline | can_offline] [, *engine_id*] |
[“shutdown”, *engine_id*]

For example, the following brings engine number one online. Messages are platform specific (in this example, Sun Solaris was used):

```
sp_engine "online", 1
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device connection
limit is 3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security modules loaded
successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 2, os pid 8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel asynchronous
disk I/O strategy
```

```
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck: Network fc0330c8 online
```

You can check whether or not a specific engine can be brought offline with the *can_offline* parameter. The following determines whether engine 1 can be brought offline:

```
sp_engine can_offline, 1
```

sp_engine specifies a return code of 0 if you can bring the specified engine offline. If you do not specify an *engine_id*, sp_engine describes the status of the engine in sysengines with the highest *engine_id*.

Engines can be brought online only if max online engines is greater then the current number of engines with an online status, and if enough CPU is available to support the additional engine.

To bring an engine offline, enter the engine ID. The following takes engine number one offline:

```
sp_engine offline, 1
```

Adaptive Server waits for any tasks that are associated with this engine to finish before taking the engine offline, and returns a message similar to the following:

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting for affinitated
process(es) before going offline
01:00000:00000:2001/11/09 16:11:11.85 kernel  Process 917518 is preventing
engine 1 going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid 21127  offline
```

You cannot take engine number zero offline.

sp_engine "shutdown" forces any tasks associated with the specified engine to finish in a five-second period, and then shuts down the engine. The following shuts down engine number one:

```
sp_engine "shutdown", 1
```

For more information about sp_engine, see the *Reference Manual*.

Relationship between network connections and engines

Due to the operating system limit on the number of file descriptors per process, reducing the number of engines reduces the number of network connections that the server can have.

There is no way to migrate a network connection created for server-to-server remote procedure calls, for example, connections to Replication Server and XP Server, so you cannot take an engine offline that is managing one of these connections.

Logical process management and *dbcc engine(offline)*

If you are using logical process management to bind particular logins or applications to engine groups, use dbcc engine(offline) carefully. If you offline all engines for an engine group:

- The login or application can run on any engine
- An advisory message is sent to the connection logging in to the server

Since engine affinity is assigned when a client logs in, users who are already logged in are not migrated if the engines in the engine group are brought online again with dbcc engine("online").

Monitoring CPU usage

To maintain the correct number of engines, monitor CPU usage with an operating system utility. See the configuration documentation for your platform for the appropriate utility for your operating system.

Taking engines offline with *dbcc engine*

You can dynamically change the number of engines in use by Adaptive Server with the dbcc engine command to take an engine offline or bring an engine online. This allows a System Administrator to reconfigure CPU resources as processing requirements fluctuate over time.

Two configuration parameters limit the number of engines available to the server:

- max online engines – when the server is started, the number of engines specified by max online engines are started. The number of engines can never exceed max online engines.
- min online engines – sets the minimum number of engines. When you take engines offline using dbcc engine, you cannot reduce the number of engines below the value set by min online engines.

Due to the operating system limit on the number of file descriptors per process, reducing the number of engines reduces the number of network connections that the server can have.

There is no way to migrate a network connection created for server-to-server remote procedure calls, for example, connections to Replication Server and XP Server, so you cannot take an engine offline that is managing one of these connections.

dbcc engine syntax and usage

The syntax for dbcc engine is:

```
dbcc engine(offline , [enginenum] )
dbcc engine("online")
```

If *enginenum* is not specified, the highest-numbered engine is taken offline.

Depending on your operating system and the load on Adaptive Server, taking an engine offline can take several minutes. To complete the task of taking an engine offline, the following steps must be completed:

- All outstanding I/Os for the engine must complete.
- All tasks affiliated with the engine must be migrated to other engines.
- Operating system and internal cleanup must de-allocate all structures.

If tasks cannot be migrated within approximately 5 minutes, the tasks are killed.

Warning! If you use dbcc engine(offline) when CPU utilization is high on the server, Adaptive Server may not be able to migrate all tasks before the time limit expires.Tasks that cannot be migrated within the time limit are killed.

Status and messages during *dbcc engine(offline)*

When a System Administrator issues a dbcc engine(offline) command, messages are sent to the error log. For example, these are the messages on Sun Solaris:

```
00:00000:00000:1999/04/08 15:09:01.13
kernel  engine 5, os pid 19441  offline
```

dbcc engine(offline) returns immediately; you must monitor the error log or check the engine status in sysengines to know that the offline-engine task completes.

An engine with open network connections using Client Library cannot be taken offline. Attempting to offline the engine reports this message in the error log:

```
00:00000:00000:1999/04/08 15:30:42.47 kernel
ueoffline: engine 3 has outstanding ct-lib
connections and cannot be offlined.
```

If there are tasks that cannot be migrated to another engine within several minutes, the task is killed, and a message similar to this is sent to the error log:

```
00:00000:00000:1999/04/08 15:20:31.26 kernel
Process 57 is killed due to engine offline.
```

Monitoring engine status

Values in the status column of sysengines track the progress of dbcc engine commands:

- online – indicates the engine is online.
- in offline – indicates that dbcc engine(offline) has been run. The engine is still allocated to the server, but is in the process of having its tasks migrated to other engines.
- in destroy – indicates that all tasks have successfully migrated off the engine, and that the server is waiting on the OS level task to deallocate the engine.
- in create – indicates that an engine is in the process of being brought online.

The following command shows the engine number, status, number of tasks affinitied, and the time an engine was brought online:

```
select engine, status, affinitied, starttime
from sysengines
```

```
engine status       affinitied  starttime
------ ------------ ----------- --------------------------
     0 online                12        Mar  5 1999  9:40PM
     1 online                 9        Mar  5 1999  9:41PM
     2 online                12        Mar  5 1999  9:41PM
     3 online                14        Mar  5 1999  9:51PM
     4 online                 8        Mar  5 1999  9:51PM
     5 in offline            10        Mar  5 1999  9:51PM
```

Logical process management and *dbcc engine(offline)*

If you are using logical process management to bind particular logins or applications to engine groups, use dbcc engine(offline) carefully. If you offline all engines for an engine group:

- The login or application can run on any engine
- An advisory message is sent to the connection logging in to the server

Since engine affinity is assigned when a client logs in, users who are already logged in are not migrated if the engines in the engine group are brought online again with dbcc engine("online").

Monitoring CPU usage

To maintain the correct number of engines, monitor CPU usage with an operating system utility. See the configuration documentation for your platform for the appropriate utility for your operating system.

Managing user connections

If the SMP system supports network affinity migration, each engine handles the network I/O for its connections. During login, Adaptive Server migrates the client connection task from engine 0 to the engine currently servicing the smallest number of connections. The client's tasks run network I/O on that engine (*network affinity*) until the connection is terminated. To determine if your SMP system supports this migration, see the configuration documentation for your platform.

By distributing the network I/O among its engines, Adaptive Server can handle more user connections. The per-process limit on the maximum number of open file descriptors no longer limits the number of connections. Adding more engines linearly increases the maximum number of file descriptors, as stored in the global variable *@@max_connections*.

As you increase the number of engines, Adaptive Server prints the increased *@@max_connections* value to standard output and the error log file after you restart the server. You can query the value as follows:

```
select @@max_connections
```

This number represents the maximum number of file descriptors allowed by the operating system for your process, minus these file descriptors used by Adaptive Server:

- One for each master network listener on engine 0 (one for every "master" line in the *interfaces* file entry for that Adaptive Server)
- One for each engine's standard output
- One for each engine's error log file
- Two for each engine's network affinity migration channel
- One per engine for configuration
- One per engine for the *interfaces* file

For example, if Adaptive Server is configured for one engine, and the value of *@@max_connections* equals 1019, adding a second engine increases the value of *@@max_connections* to 2039 (assuming only one master network listener).

You can configure the number of user connections parameter to take advantage of an increased *@@max_connections* limit. However, each time you decrease the number of engines using max online engines, you must also adjust the number of user connections value accordingly. Reconfiguring max online engines or number of user connections is not dynamic, so you must restart the server to change these configuration values. For information about configuring number of user connections, see Chapter 4, "Setting Configuration Parameters."

Configuration parameters that affect SMP systems

Chapter 4, "Setting Configuration Parameters," lists configuration parameters for Adaptive Server. Some of those parameters, such as spinlock ratios, are applicable only to SMP systems.

Configuring spinlock ratio parameters

Spinlock ratio parameters specify the number of internal system resources such as rows in an internal table or cache that are protected by one spiinlock. A spinlock is a simple locking mechanism that prevents a process from accessing the system resource currently used by another process. All processes trying to access the resource must wait (or "spin") until the lock is released.

Spinlock ratio configuration parameters are meaningful only in multiprocessing systems. An Adaptive Server configured with only one engine has only one spinlock, regardless of the value specified for a spinlock ratio configuration parameter.

Table 20-1 lists system resources protected by spinlocks and the configuration parameters you can use to change the default spinlock ratio.

Table 20-1: Spinlock ratio configuration parameters

Configuration parameter	System resource protected
lock spinlock ratio	Number of lock hash buckets
open index hash spinlock ratio	Index metadata descriptor hash tables
open index spinlock ratio	Index metadata descriptors
open object spinlock ratio	Object metadata descriptors
partition spinlock ratio	Rows in the internal partition caches
user log cache spinlock ratio	User log caches

The value specified for a spinlock ratio parameter defines the ratio of the particular resource to spinlocks, not the number of spinlocks. For example, if 100 is specified for the spinlock ratio, Adaptive Server allocates one spinlock for each 100 resources. The number of spinlocks allocated by Adaptive Server depends on the total number of resources as well as on the ratio specified. The lower the value specified for the spinlock ratio, the higher the number of spinlocks.

Spinlocks are assigned to system resources in one of two ways:

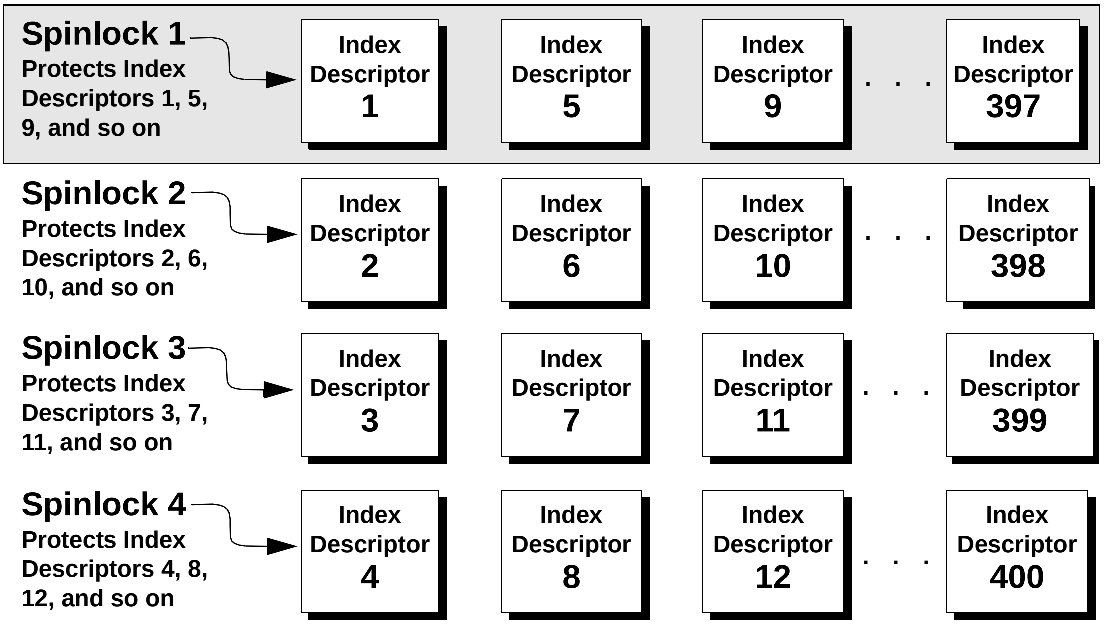
- Round-robin assignment
- Sequential assignment

Round-robin assignment

Metadata cache spinlocks (configured by the open index hash spinlock ratio, open index spinlock ratio, and open object spinlock ratio parameters) use the round-robin assignment method.

Figure 20-2 illustrates one example of the round-robin assignment method and shows the relationship between spinlocks and index metadata descriptors.

Figure 20-2: Relationship between spinlocks and index descriptors



Suppose there are 400 index metadata descriptors, or 400 rows in the index descriptors internal table. You have set the ratio to 100. This means that there will be 4 spinlocks in all: Spinlock 1 protects row 1; Spinlock 2 protects row 2, Spinlock 3 protects row 3, and Spinlock 4 protects row 4. After that, Spinlock 1 protects the next available index descriptor, Index Descriptor 5, until every index descriptor is protected by a spinlock. This round-robin method of descriptor assignment reduces the chances of spinlock contention.

Sequential assignment

Table lock spinlocks, configured by the table lock spinlock ratio parameter, use the sequential assignment method. The default configuration for table lock spinlock ratio is 20, which assigns 20 rows in an internal hash table to each spinlock. The rows are divided up sequentially: the first spinlock protects the first 20 rows, the second spinlock protects the second 20 rows, and so on.

In theory, protecting one resource with one spinlock would provide the least contention for a spinlock and would result in the highest concurrency. In most cases, the default value for these spinlock ratios is probably best for your system. Change the ratio only if there is spinlock contention.

Use sp_sysmon to get a report on spinlock contention. See the *Performance and Tuning Guide* for information on spinlock contention.

CHAPTER 21

Creating and Managing User Databases

This chapter explains how to create and manage user databases.

Commands for creating and managing user databases

Table 21-1 summarizes the commands for creating, modifying, and dropping user databases and their transaction logs.

Table 21-1: Commands for managing user databases

Command	Task
create database...on *dev_name* or alter database...on *dev_name*	Makes database devices available to a particular Adaptive Server database. When used without the on *dev_name* clause, these commands allocate space from the default pool of database devices.
dbcc checktable(syslogs)	Reports the size of the log.
sp_logdevice	Specifies a device that will store the log when the current log device becomes full.
sp_helpdb	Reports information about a database's size and devices.
sp_spaceused	Reports a summary of the amount of storage space used by a database.

Permissions for managing user databases

By default, only the System Administrator has create database permission. The System Administrator can grant permission to use the create database command. However, in many installations, the System Administrator maintains a monopoly on create database permission to centralize control of database placement and database device allocation. In these situations, the System Administrator creates new databases on behalf of other users and then transfers ownership to the appropriate user(s).

To create a database and transfer ownership to another user, the System Administrator:

1. Issues the create database command.
2. Switches to the new database with the use *database* command.
3. Executes sp_changedbowner, as described in “Changing database ownership” on page 684.

When a System Administrator grant permission to create databases, the user that receives the permission must also be a valid user of the master database, since all databases are created while using master.

The fact that System Administrators seem to operate outside the protection system serves as a safety precaution. For example, if a Database Owner forgets his or her password or accidentally deletes all entries in sysusers, a System Administrator can repair the damage using the backups or dumps that are made regularly.

Permission alter database or drop database defaults to the Database Owner, and permission is automatically transferred with database ownership. alter database and drop database permission cannot be changed with grant or revoke.

Using the *create database* command

Use create database to create user databases. You must have create database permission, and you must be a valid user of master. Always type use master before you create a new database.

Note Each time you enter the create database command, dump the master database. This makes recovery easier and safer in case master is later damaged. See Chapter 29, “Restoring the System Databases,” for more information.

---

create database syntax

The create database syntax is:

```
create database database_name
    [on {default | database_device} [= size]
        [, database_device [= size]...]
    [log on database_device [ = size ]
        [, database_device [= size]]...]
    [with {override | default_location = "pathname"}]
    [for {load | proxy_update}]
```

A database name must follow the rules for identifiers. You can create only one database at a time.

In its simplest form, create database creates a database on the default database devices listed in master..sysdevices:

```
create database newpubs
```

You can control different characteristics of the new database by using the create database clauses:

- The on clause specifies the names of one or more database devices and the space allocation, in megabytes, for each database device. See “Assigning space and devices to databases” on page 677 for more information.
- The log on clause places the **transaction log** (the syslogs table) on a separate database device with the specified or default size. See “Placing the transaction log on a separate device” on page 679 for more information.

- for load causes Adaptive Server to skip the page-clearing step during database creation. Use this clause if you intend to load a dump into the new database as the next step. See “Using the for load option for database recovery” on page 683 for more information.
- with override allows Adaptive Servers on machines with limited space to maintain their logs on device fragments that are separate from their data. Use this option *only* when you are putting log and data on the same logical device. See “Using the with override option with create database” on page 684 for more information.
- *size* is in the following unit specifiers: ‘k’ or ‘K’ (kilobytes), ‘m’ or ‘M’ (megabytes), and ‘g’ or ‘G’ (gigabytes).

How *create database* works

When a user with the required permission issues create database, Adaptive Server:

- Verifies that the database name specified is unique.
- Makes sure that the database device names specified are available.
- Finds an unused identification number for the new database.
- Assigns space to the database on the specified database devices and updates master..sysusages to reflect these assignments.
- Inserts a row into sysdatabases.
- Makes a copy of the model database in the new database space, thereby creating the new database’s system tables.
- Clears all the remaining pages in the database device. If you are creating a database to load a database dump, for load skips page clearing, which is performed after the load completes).

The new database initially contains a set of system tables with entries that describe the system tables themselves. The new database inherits all the changes you have made to the model database, including:

- The addition of user names.
- The addition of objects.

- The database option settings. Originally, the options are set to “off” in model. If you want all of your databases to inherit particular options, change the options in model with sp_dboption. See Chapter 2, “System and Optional Databases,” for more information about model. See Chapter 23, “Setting Database Options,” for more information about changing database options.

Adding users to databases

After creating a new database, the System Administrator or Database Owner can manually add users to the database with sp_adduser. This task can be done with the help of the System Security Officer, if new Adaptive Server logins are required. See Chapter 9, “Security Administration,” for details on managing Adaptive Server logins and database users.

Assigning space and devices to databases

Adaptive Server allocates storage space to databases when a user enters the create database or alter database command. create database can specify one or more database devices, along with the amount of space on each that is to be allocated to the new database.

Note You can also use the log on clause to place a production database’s transaction log on a separate device. See “Placing the transaction log on a separate device” on page 679 for more information.

If you use the default keyword, or if you omit the on clause, Adaptive Server puts the database on one or more of the default database devices specified in master..sysdevices. See “Designating default devices” on page 566 for more information about the default pool of devices.

To specify a size (4MB in the following example) for a database that is to be stored in a default location, use on default = size like this:

```
create database newpubs
on default = "4M"
```

To place the database on specific database devices, give the name(s) of the database device(s) where you want it stored. You can request that a database be stored on more than one database device, with a different amount of space on each. All the database devices named in create database must be listed in sysdevices. In other words, they must have been initialized with disk init. See Chapter 16, “Initializing Database Devices,” for instructions about using disk init.

The following statement creates the newdb database and allocates 3MB on mydata and 2MB on newdata. The database and transaction log are not separated:

```
create database newdb
on mydata = "3M", newdata = "2M"
```

Warning! Unless you are creating a small or noncritical database, always place the log on a separate database device. Follow the instructions under “Placing the transaction log on a separate device” on page 679 to create production databases.

If the amount of space you request on a specific database device is unavailable, Adaptive Server creates the database with as much space as possible on each device and displays a message informing you how much space it has allocated on each database device. This is not considered an error. If there is less than the minimum space necessary for a database on the specified database device, create database fails.

If you create (or alter) a database on a UNIX device file that does not use the dsync setting, Adaptive Server displays an error message in the error log file. For example, if you create the “mydata” device in the previous example does not use dsync, you would see a message similar to:

```
Warning: The database 'newdb' is using an unsafe virtual device 'mydata'. The
recovery of this database can not be guaranteed.
```

Default database size and devices

If you omit the size parameter in the on clause, Adaptive Server creates the database with a default amount of space. This amount is the larger of the sizes specified by the default database size configuration parameter and the model database.

The size of model and the value of default database size are initially set to the size of the server's logical page. To change the size of model, allocate more space to it with alter database. To change the default database size configuration parameter, use sp_configure. Changing default database size enables you to set the default size for new databases to any size between the server's logical page size and 10,000MB. See "default database size" on page 154 for complete instructions.

If you omit the on clause, the database is created as the default size, as described above. The space is allocated in alphabetical order by database device name, from the default database devices specified in master..sysdevices.

To see the logical names of default database devices, enter:

```
select name
    from sysdevices
    where status & 1 = 1
    order by name
```

sp_helpdevice also displays "default disk" as part of the description of database devices.

Estimating the required space

The size allocation decisions you make are important, because it is difficult to reclaim storage space after it has been assigned. You can always add space; however, you cannot de-allocate space that has been assigned to a database, unless you drop the database first.

You can estimate the size of the tables and indexes for your database by using sp_estspace or by calculating the value. See Chapter 15, "Determining Sizes of Tables and Indexes," in the *Performance and Tuning Guide* for instructions.

Placing the transaction log on a separate device

Use the log on clause of the create database command to place the transaction log (the syslogs table) on a separate database device. Unless you are creating very small, noncritical databases, always place the log on a separate database device. Placing the logs on a separate database device:

- Lets you use dump transaction, rather than dump database, thus saving time and tapes.
- Lets you establish a fixed size for the log to keep it from competing for space with other database activity.
- Creates default free-space threshold monitoring on the log segment and allows you to create additional free-space monitoring on the log and data portions of the database. See Chapter 31, “Managing Free Space with Thresholds,” for more information.
- Improves performance.
- Ensures full recovery from hard disk crashes. A special argument to dump transaction lets you dump your transaction log, even when your data device is on a damaged disk.

To specify a size and device for the transaction log, use the log on *device* = *size* clause to create database. The size is in the unit specifiers ‘k’ or ‘K’ (kilobytes), ‘m’ or ‘M’ (megabytes), and ‘g’ or ‘G’ (gigabytes). For example, the following statement creates the newdb database, allocates 8MB on mydata and 4MB on newdata, and places a 3MB transaction log on a third database device, tranlog:

```
create database newdb
on mydata = "8M", newdata = "4M"
log on tranlog = "3M"
```

Estimating the transaction log size

The size of the transaction log is determined by:

- The amount of update activity in the associated database
- The frequency of transaction log dumps

This is true whether you perform transaction log dumps manually or use threshold procedures to automate the task. As a general rule, allocate to the log 10 to 25 percent of the space that you allocate to the database.

Inserts, deletes, and updates increase the size of the log. dump transaction decreases its size by writing committed transactions to disk and removing them from the log. Since update statements require logging the "before" and "after" images of a row, applications that update many rows at once should plan on the transaction log being at least twice as large as the number of rows to be updated at the same time, or twice as large as your largest table. Or you can **batch** the updates in smaller groups, performing transaction dumps between the batches.

In databases that have a lot of insert and update activity, logs can grow very quickly. To determine the required log size, periodically check the size of the log. This will also help you choose thresholds for the log and scheduling the timing of transaction log dumps. To check the space used by a database's transaction log, first use the database. Then enter:

```
dbcc checktable(syslogs)
```

dbcc reports the number of data pages being used by the log. If your log is on a separate device, dbcc checktable also tells you how much space is used and how much is free. Here is sample output for a 2MB log:

```
Checking syslogs
The total number of data pages in this table is 199.
*** NOTICE: Space used on the log segment is 0.39 Mbytes, 19.43%.
*** NOTICE: Space free on the log segment is 1.61 Mbytes, 80.57%.
Table has 1661 data rows.
```

You can also use the following Transact-SQL statement to check on the growth of the log:

```
select count(*) from syslogs
```

Repeat either command periodically to see how fast the log grows.

Default log size and device

If you omit the *size* parameter in the log on clause, Adaptive Server allocates one logical page of storage on the specified log device. If you omit the log on clause entirely, Adaptive Server places the transaction log on the same database device as the data tables.

Moving the transaction log to another device

If you did not use the log on clause to create database, follow the instructions in this section to move your transaction log to another database device.

sp_logdevice moves the transaction log of a database with log and data on the same device to a separate database device. However, the transaction log remains on the original device until the allocated page has been filled and the transaction log has been dumped.

Note If the log and its database share the same device, subsequent use of sp_logdevice affects only future writes to the log; it does not immediately move the first few log pages that were written when the database was created. This creates exposure problems in certain recovery situations, and is not recommended.

The syntax for sp_logdevice is:

sp_logdevice *database_name*, *devname*

The database device you name must be initialized with disk init and must be allocated to the database with create or alter database.

To move the entire transaction log to another device:

1. Execute sp_logdevice, naming the new database device.
2. Execute enough transactions to fill the page that is currently in use. The amount of space you will need to update depends on the size of your logical pages. You can execute dbcc checktable(syslogs) before and after you start updating to determine when a new page is used.
3. Wait for all currently active transactions to finish. You may want to put the database into single-user mode with sp_dboption.
4. Run dump transaction, which removes all the log pages that it writes to disk. As long as there are no active transactions in the part of the log on the old device, all of those pages will be removed. See Chapter 27, “Developing a Backup and Recovery Plan,” for more information.

5 Run sp_helplog to ensure that the complete log is on the new log device.

Note When you move a transaction log, the space no longer used by the transaction log becomes available for data. However, you cannot reduce the amount of space allocated to a device by moving the transaction log.

Transaction logs are discussed in detail in Chapter 27, “Developing a Backup and Recovery Plan.”

Using the *for load* option for database recovery

Adaptive Server generally clears all unused pages in the database device when you create a new database. Clearing the pages can take several seconds or several minutes to complete, depending on the size of the database and the speed of your system.

Use the for load option if you are going to use the database for loading from a database dump, either for recovery from media failure or for moving a database from one machine to another. Using for load runs a streamlined version of create database that skips the page-clearing step, and creates a target database that can be used *only* for loading a dump.

If you create a database using for load, you can run only the following commands in the new database before loading a database dump:

- alter database...for load
- drop database
- load database

When you load a database dump, the new database device allocations for the database need to match the usage allocations in the dumped database. See Chapter 28, “Backing Up and Restoring User Databases,” for a discussion of duplicating space allocation.

After you load the database dump into the new database, there are no restrictions on the commands you can use.

Using the *with override* option with *create database*

This option allows machines with limited space to maintain their logs on device fragments that are separate from their data. Use this option *only* when you put log and data on the same logical device. Although this is not recommended practice, it may be the only option available on machines with limited storage, especially if you need to get databases back online following a hard disk crash.

You will still be able to dump your transaction log, but if you experience a media failure, you will not be able to access the current log, since it is on the same device as the data. You will be able to recover only to the last transaction log dump, and all transactions between that point and the failure time will be lost.

In the following example, the log and data are on separate fragments of the same logical device:

```
create database littledb
    on diskdev1 = "4M"
    log on diskdev1 = "1M"
    with override
```

The minimum database size you can create is the size of model.

Changing database ownership

A System Administrator might want to create the user databases and give ownership of them to another user after completing some of the initial work. sp_changedbowner changes the ownership of a database. The procedure must be executed by the System Administrator in the database where the ownership will be changed. The syntax is:

sp_changedbowner *loginame* [, true]

The following example makes the user “albert” the owner of the current database and drops the aliases of users who could act as the former “dbo.”

```
sp_changedbowner albert
```

The new owner must already have a login name in Adaptive Server, but he or she 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. See the Chapter 9, "Security Administration," for more information about changing ownership.

To transfer aliases and their permissions to the new Database Owner, add the second parameter, true.

Note You cannot change ownership of the master database. It is always owned by the "sa" login.

Using the *alter database* command

When your database or transaction log grows to fill all the space allocated with create database, you can use alter database to add storage. You can add space for database objects or the transaction log, or both. You can also use alter database to prepare to load a database from backup.

Permission for alter database defaults to the Database Owner, and is automatically transferred with database ownership. For more information, see "Changing database ownership" on page 684. alter database permission cannot be changed with grant or revoke.

Note alter database for proxy update drops all proxy tables in the proxy database.

---

alter database syntax

To extend a database, and to specify where storage space is to be added, use the full alter database syntax:

```
alter database database_name
    [on {default | database_device} [= size]
        [, database_device [= size]]...]
    [log on {default | database_device} [= size]
        [, database_device [= size]]...]
    [with override]
```

[for load]
[for proxy_update]

In its simplest form, alter database adds one logical page from the default database devices. If your database separates log and data, the space you add is used only for data. Use sp_helpdevice to find names of database devices that are in your default list.

To add logical page from a default database device to the newpubs database, enter:

```
alter database newpubs
```

The on and log on clauses operate like the corresponding clauses in create database. You can specify space on a default database device or some other database device, and you can name more than one database device. If you use alter database to extend the master database, you can extend it only on the master device. The minimum increase you can specify is 1MB or one allocation unit, whichever is larger.

To add 3MB to the space allocated for the newpubs database on the database device named pubsdata1, enter:

```
alter database newpubs
on pubsdata1 = "3M"
```

If Adaptive Server cannot allocate the requested size, it allocates as much as it can on each database device, with a minimum allocation of .5MB (256 2K pages) per device. When alter database completes, it prints messages telling you how much space it allocated; for example:

```
Extending database by 1536 pages on disk pubsdata1
```

Check all messages to make sure the requested amount of space was added.

The following command adds 2MB to the space allocated for newpubs on pubsdata1, 3MB on a new device, pubsdata2, and 1MB for the log on tranlog:

```
alter database newpubs
on pubsdata1 = "2M", pubsdata2 =" 3M"
log on tranlog
```

Note Each time you issue the alter database command, dump the master database.

Use with override to create a device fragment containing log space on a device that already contains data or a data fragment on a device already in use for the log. Use this option only when you have no other storage options and when up-to-the-minute recoverability is not critical.

Use for load only after using create database for load to re-create the space allocation of the database being loaded into the new database from a dump. See Chapter 28, “Backing Up and Restoring User Databases,” for a discussion of duplicating space allocation when loading a dump into a new database.

Using the *drop database* command

Use drop database to remove a database from Adaptive Server, thus deleting the database and all the objects in it. This command:

- Frees the storage space allocated for the database
- Deletes references to the database from the system tables in the master database

Only the Database Owner can drop a database. You must be in the master database to drop a database. You cannot drop a database that is open for reading or writing by a user.

The syntax is:

drop database *database_name* [, *database_name*]...

You can drop more than one database in a single statement; for example:

```
drop database newpubs, newdb
```

You must drop all databases from a database device before you can drop the database device itself. The command to drop a device is sp_dropdevice.

After you drop a database, dump the master database to ensure recovery in case master is damaged.

System tables that manage space allocation

To create a database on a database device and allocate a certain amount of space to it, Adaptive Server first makes an entry for the new database in sysdatabases. Then, it checks master..sysdevices to make sure that the device names specified in create database actually exist and are database devices. If you did not specify database devices, or used the default option, Adaptive Server checks master..sysdevices and master..sysusages for free space on all devices that can be used for default storage. It performs this check in alphabetical order by device name.

The storage space from which Adaptive Server gathers the specified amount of storage need not be contiguous. The database storage space can even be drawn from more than one database device. A database is treated as a logical unit, even if it is stored on more than one database device.

Each piece of storage for a database must be at least 1 allocation unit. The first page of each allocation unit is the allocation page. It does not contain database rows like the other pages, but contains an array that shows how the rest of the pages are used.

The *sysusages* table

The database storage information is listed in master..sysusages. Each row in master..sysusages represents a space allocation assigned to a database. Thus, each database has one row in sysusages for each time create database or alter database assigns a fragment of disk space to it.

When you install Adaptive Server, sysusages contains rows for these dbids:

- 1, the master database
- 2, the temporary database, tempdb
- 3, the model database
- 4, the sybsystemprocs database

If you installed auditing, the sybsecurity database will be dbid 5.

As new databases are created or current databases enlarged, new rows are added to sysusages to represent new database allocations.

Here is what sysusages might look like on an Adaptive Server with the five system databases and two user databases (with dbids 6 and 7). Both user databases were created with the log on option. The database with dbid 7 has been given additional storage space with two alter database commands:

```
                                    select dbid, segmap, lstart, size, vstart
                                    from sysusages
dbid   segmap      lstart      size            vstart
------ ----------- ----------- -----------     -------
     1           7           0        1536           4
     2           7           0        1024        2564
     3           7           0        1024        1540
     4           7           0        5120    16777216
     5           7           0       10240    33554432
     6           3           0         512     1777216
     6           4         512         512     3554432
     7           3           0        2048    67108864
     7           4        2048        1024    50331648
     7           3        3072         512    67110912
     7           3        3584        1024    67111424

(10 rows affected)
```

The *segmap* column

The segmap column is a bitmask linked to the segment column in the user database's syssegments table. Since the logsegment in each user database is segment 2, and these user databases have their logs on separate devices, segmap contains 4 (2^2) for the devices named in the log on statement and 3 for the data segment that holds the system segment ($2^0 = 1$) + default segment ($2^1 = 2$).

Some possible values for segments containing data or logs are:

Value	Segment
3	Data only (system and default segments)
4	Log only
7	Data and log

Values higher than 7 indicate user-defined segments. The segmap column is explained more fully in the segments tutorial section in Chapter 24, “Creating and Using Segments.”

The *lstart*, *size*, and *vstart* columns

- lstart column – the starting page number in the database of this allocation unit. Each database starts at logical address 0. If additional allocations have been made for a database, as in the case of dbid 7, the lstart column reflects this.
- size column – the number of contiguous pages that are assigned to the same database. The ending logical address of this portion of the database can be determined by adding the values in lstart and size.
- vstart column – the address where the piece assigned to this database begins. The upper 4 bits store the virtual device number (vdevno), and the lower 4 bits store the virtual block number. (To obtain the virtual device number, divide sysusages.vstart or sysdevices.low by 16,777,216, which is 2^{24}.) The value in vstart identifies which database device contains the page number of the database, because it falls between the values in the low and high columns of sysdevices for the database device in question.

Getting information about database storage

This section explains how to determine which database devices are currently allocated to databases and how much space each database uses.

Database device names and options

To find the names of the database devices on which a particular database resides, use sp_helpdb with the database name:

```
                           sp_helpdb pubs2
name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     2.0 MB     sa           5 Aug 25, 1997   no options set

device_fragments    size          usage           free kbytes
------------------- ------------- --------------- -----------
pubdev              2.0 MB        data and log            288

device                 segment
---------------------- ----------------------
```

```
pubdev                  default
pubdev                  logsegment
pubdev                  system
```

sp_helpdb reports on the size and usage of the devices used by the named database. The status column lists the database options. These options are described in Chapter 23, “Setting Database Options.”

If you are using the named database, sp_helpdb also reports on the segments in the database and the devices named by the segments. See Chapter 24, “Creating and Using Segments,” for more information.

When you use sp_helpdb without arguments, it reports information about all databases in Adaptive Server:

```
                                sp_helpdb
name           db_size  owner dbid created      status
------------- -------- ----- ---- ----------- -------------------
master           3.0 MB sa       1 Jan 01, 1900 no options set
model            2.0 MB sa       3 Jan 01, 1900 no options set
mydata           4.0 MB sa       7 Aug 25, 1997 no options set
pubs2            2.0 MB sa       6 Aug 23, 1997 no options set
sybsecurity     20.0 MB sa       5 Aug 18, 1997 no options set
sybsystemprocs 10.0 MB sa       4 Aug 18, 1997 trunc log on chkpt
tempdb           2.0 MB sa       2 Aug 18, 1997 select into/
                                                bulkcopy/pllsort
```

Checking the amount of space used

sp_spaceused provides:

- A summary of space used in the database
- A summary of space used by a table and its indexes and text/image storage
- A summary of space used by a table, with separate information on indexes and text/image storage.

Checking space used in a database

To get a summary of the amount of storage space used by a database, execute sp_spaceused in the database:

```
sp_spaceused
database_name                  database_size
------------------------------ -------------
pubs2                          2.0 MB

reserved      data          index_size      unused
------------- ------------- --------------- --------
-
1720 KB       536 KB        344 KB          840 KB
```

Table 21-2 describes the columns in the report.

Table 21-2: Columns in sp_spaceused output

Column	Description
database_name	The name of the database being examined.
database_size	The amount of space allocated to the database by create database or alter database.
reserved	The amount of space that has been allocated to all the tables and indexes created in the database. (Space is allocated to database objects inside a database in increments of 1 extent, or 8 pages, at a time.)
data, index_size	The amount of space used by data and indexes.
unused	The amount of space that has been reserved but not yet used by existing tables and indexes.

The sum of the values in the unused, index_size, and data columns should equal the figure in the reserved column. Subtract reserved from database_size to get the amount of unreserved space. This space is available for new or existing objects that grow beyond the space that has been reserved for them.

By running sp_spaceused regularly, you can monitor the amount of database space available. For example, if the reserved value is close to the database_size value, you are running out of space for new objects. If the unused value is also small, you are running out of space for additional data as well.

Checking summary information for a table

You can also use sp_spaceused with a table name as its parameter:

```
sp_spaceused titles
name   rowtotal reserved  data    index_size unused
------ -------- --------- ------- ---------- -----
titles 18       48 KB     6 KB    4 KB       38 KB
```

The rowtotal column may be different than the results of running select count(*) on the table. This is because sp_spaceused computes the value with the built-in function rowcnt. That function uses values that are stored in the allocation pages. These values are not updated regularly, however, so they can be very different for tables with a lot of activity. update statistics, dbcc checktable, and dbcc checkdb update the rows-per-page estimate, so rowtotal will be most accurate after you have run one of these commands has been run.

You should run sp_spaceused regularly on syslogs, since the transaction log can grow rapidly if there are frequent database modifications. This is particularly a problem if the transaction log is not on a separate device—in which case, it competes with the rest of the database for space.

Checking information for a table and its indexes

To see information on the space used by individual indexes, enter:

```
                               sp_spaceused titles, 1
index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
titleidind           2 KB       32 KB      24 KB
titleind             2 KB       16 KB      14 KB

name       rowtotal   reserved   data   index_size unused
---------- --------  ---------   ------- ---------- ----------
titles          18      46 KB     6 KB       4 KB      36 KB
```

Space taken up by the text/image page storage is reported separately from the space used by the table. The object name for text/image storage is always “t” plus the table name:

```
                               sp_spaceused blurbs,1
index_name           size       reserved   unused
-------------------- ---------- ---------- ----------
blurbs               0 KB       14 KB      12 KB
tblurbs              14 KB      16 KB      2 KB

name       rowtotal reserved    data    index_size unused
---------- -------- ---------- ------- ---------- ----------
blurbs            6       30 KB    2 KB      14 KB      14 KB
```

Querying system table for space usage information

You may want to write some of your own queries for additional information about physical storage. For example, to determine the total number of 2K blocks of storage space that exist on Adaptive Server, you can query sysdevices:

```
select sum(high - low)
from sysdevices
where status in (2, 3)
```

```
-------------------
               7168
```

A 2 in the status column represents a physical device; a 3 represents a physical device that is also a default device.

CHAPTER 22 Database Mount and Unmount

The mount and unmount commands make it possible to transport databases from a source Adaptive Server to a destination Adaptive Server.

The key benefits of the mount and unmount commands are:

1 The copy operation provided by the quiesce and mount commands makes it possible to package proprietary databases more easily; for example, data files instead of SQL scripts. The associated actions necessary for running these SQL scripts, such as device and database setup, are eliminated.

2 The quiesce database extension for mount makes it possible to copy databases without a shutdown of the Adaptive Server. The primary Adaptive Server is the source, and the secondary Adaptive Server is the destination. quiesce database also allows a single secondary to act as standby for databases from multiple primaries, since databases from multiple sources can be copied to a single destination.

The mount and unmount commands support two operations:

- Copying a database

  When you copy a database you must use a command outside of Adaptive Server, such as UNIX dd or ftp, to create a byte-for-byte copy of all pages in a set of one or more databases.

- Moving a database

  When you move a set of databases from a source Adaptive Server to a destination Adaptive Server, you are physically moving the underlying devices.

With the mount and unmount commands:

1. Remove a database using unmount, which removes a database and its devices from a server. A manifest file is created for the database that you are unmounting at a location you specify in the command clauses. The manifest file contains information pertinent to the database at the source Adaptive Server, such as database devices, server information, and database information. For more information see "Manifest file" on page 696.
2. Copy or move the database onto the destination Adaptive Server, then use mount to add the devices, attributes, and so on for the database.
3. Use database online to bring the database up on the destination Adaptive Server without rebooting the server.

Warning! For every login that is allowed access to a database on the original Adaptive Server, a corresponding login for the same suid must exist at the destination Adaptive Server.

For permissions to remain unchanged, the login maps at the destination Adaptive Server must be identical to those on the source Adaptive Server.

Manifest file

The manifest file is a binary file that contains information pertinent to the database, such as database devices, server information, and database information. Both the source and destination Adaptive Servers must use the same version of the manifest file. It can be created only if the set of databases that occupy those devices are isolated and self contained. The manifest file contains:

- Source server information that is server-specific or common to all the databases, such as the version of the source Adaptive Server, any upgrade version, pagesize, charset, sort order, and the date the manifest file was created.
- Device information that is derived from the *sysdevices* catalog. The manifest file contains the information on the first and last virtual pages, the status, and the logical and the physical names of the devices involved.

- Database information that is derived from the *sysdatabases* catalog. The manifest file contains information on the database name, dbid, login name of the database administrator (dba), suid of the owner, pointer to the transaction log, creation date, status bits from the status fields in *sysdatabases*, date of the last dump tran, and the diskmap entries of the databases involved.

Warning! The manifest file is a binary file, so operations that perform character translations of the file contents (such as ftp) corrupt the file, unless performed in binary mode.

Considerations

There are several things you should consider in using the mount and unmount commands.

Device allocation

Moving or copying are database level operations that require activity that is external to Adaptive Server. To do the move or copy of the devices and databases, you need be sure that they are setup on the source Adaptive Server in units supporting a physical transportation.

For example, if any device is used by more than one database, then all of those databases have to be transported in one operation.

Moving a database

Moving a set of one or more databases means physically moving the underlying devices. unmount the databases at the source, it creates the manifest file, and move the devices to the destination Adaptive Server.

At the destination Adaptive Server, mount the database using the manifest file. To activate the database, use the online database command. You need not reboot the destination server.

Use caution during the initial configuration of the source Adaptive Server. The Adaptive Server cannot know if a device is transportable as a unit. Make sure that the underlying disk that is to be disconnected will not cause a database that is not being moved to lose some of its storage. Adaptive Server cannot identify whether a drive is partitioned on a single physical disk; you must move the databases together in one unit.

Warning! mount and unmount allow you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

Copying a database

Copying is the process of duplicating a set of databases from the source to a destination by physically copying the underlying devices. In this case, you are copying a set of databases from a source Adaptive Server to a destination Adaptive Server.

The quiesce database for an external dump includes the extension for creating the manifest file, then use a utility or command external to Adaptive Server, (tar, zip or the UNIX dd command) to move or copy the database to the destination Adaptive Server. The datadump and the manifest file are then shipped to the destination Adaptive Server. Data is extracted and placed on devices at the destination Adaptive Server by the tar, unzip or dd command.

If a device is used for more than one database, all of the databases on that device must be moved in one operation.

Performance considerations

- Database IDs for the transported databases must be the same on the destination Adaptive Server, otherwise mount displays a message asking that dbcc checkalloc be run on the database.

  You must run checkalloc to fix up the database ID if the database is not being mounted for temporary usage.

- If the dbid is changed, all stored procedures are marked for recompiling in the database. This increases the time taken to recover the database at the destination and delays the first execution of the procedure.

System restrictions

- You cannot unmount system databases. However, you can unmount sybsystemprocs.
- You cannot unmount proxy databases or user created temporary databases.
- You cannot use the mount and unmount commands in a transaction.
- You cannot mount a database on an HA-configured server.
- A mount or unmount command limits the number of databases to eight in a single command.

Configuration restrictions

When executing the mount command at the destination Adaptive Server, use these rules:

- The page size in the destination Adaptive Server must be equal to the page size in the source Adaptive Server.
- There must be enough devices configured at the destination Adaptive Server to be able to successfully add all the devices belonging to the mounted databases.
- You cannot have databases and devices with the same names on the destination Adaptive Server as those being mounted from the source Adaptive Server.
- The destination Adaptive Server and the source Adaptive Server must use the same version as the manifest file.
- The platforms of the source and destination Adaptive Servers must be the same.
- Differences in the sort order or charset are resolved by rules followed by load database. A database with a different charset can be mounted only if the sort order is binary.

Logical restrictions

You cannot mount a subset of a transported set of databases at the destination Adaptive Server; all databases and their devices in the manifest file must be mounted together.

A set of databases copied to a destination Adaptive Server cannot contain any references to databases outside of the set.

Device verification

The destination Adaptive Server verifies the set of devices provided in the manifest file by scanning the device allocation boundaries for each database. The scan ensures that the devices being mounted correspond to the allocations described in the manifest file and verifies the dbid in the allocation page against that in the manifest file for the first and last allocation pages for each *sysusages* entry.

If a stricter device inspection is necessary, you can use the with verify in the mount command, which verifies the dbid for all allocation pages in the databases.

You must exercise extreme care so as not to mix up copies of the devices.

For example, if you make a database copy made up of copies for disks dsk1, dsk2 and dsk3:

In August, you try to mount dsk1 and dsk2 copies from a copy of the database made in March, and dsk3 from a copy made in June, the allocation page check passes even if with verify is used in the mount command. The database recovery fails since the information is not valid for the version being used.

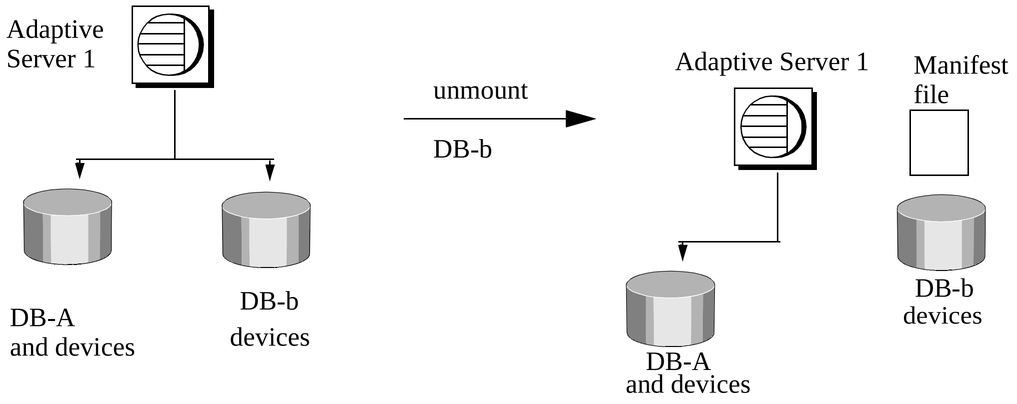
However, recovery may not fail if dsk3 is not accessed, which means that the database will come online but the data could be corrupt.

Commands

This section explains how the mount and unmount commands are used. A new clause that facilitates mount and unmount commands has been added to the quiesce database command.

unmounting a database

Figure 22-1: unmount command



Note unmount allows you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

When you unmount a database, you remove the database and its devices from an Adaptive Server. The unmount command shuts down the database. All tasks using the database are terminated. The database and its pages are not altered and remain on the OS devices.

The unmount command limits the number of databases that can be moved in a single command to eight.

The unmount command:

- Shuts down the database,
- Drops the database from the Adaptive Server,
- Devices are deactivated and dropped,
- Use the *manifest_ file* extension to create the manifest file.

Once the unmount command completes, you can disconnect and move the devices at the source Adaptive Server if necessary.

```
unmount database <dbname list> to <manifest_file> [with
{override, [waitfor=<delay time]} ]
```

For example:

```
1> unmount database pubs2 to
"/work2/Devices/Mpubs2_file"
2> go
```

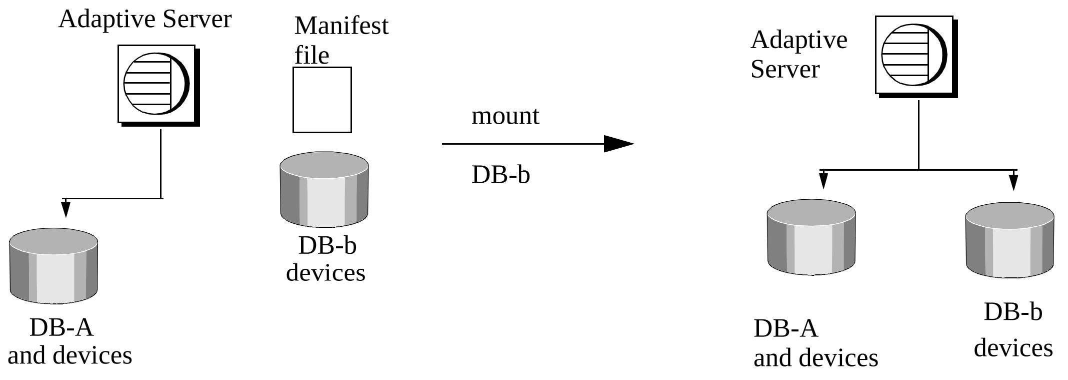
If you now try to use the pubs2 database, you see this error:

```
Attempt to locate entry in sysdatabases for database
'pubs2' by name failed - no entry found under that name.
Make sure that name is entered properly.
```

Note When the referencing database is dropped by the unmount command with an override, you cannot drop the referential constraints (dependencies) or table.

Mounting a database

Figure 22-2: mount command



Use the mount command to attach the database to the destination or secondary Adaptive Server. The mount command decodes the information in the manifest file and makes the set of databases available online. All the required supporting activities are executed, including adding database devices, if necessary, and activating them, creating the catalog entries for the new databases, recovering them, and putting them online.

The mount command limits the number of databases to eight in a single command.

Note mount allows you to identify more than one database for a move operation. However, if a device is used for more than one database, then all of the databases must be moved in one operation. You specify the set of databases being transported. The devices used by these databases cannot be shared with any extraneous database besides the ones specified in the command.

You can use the mount command in different ways:

- When you are ready, use the mount command at the destination Adaptive Server:

  mount database all from <*manifest_file*> [with {verify, listonly}] [using *device_specifications*]

  device_specifications ::= ”physicalname”[=”device_name”][,”physicalname”[‘device_name]]

  For example:

  ```
  1> mount database all from "/data/sybase2/mfile1”
  using “/data/sybase1/d0.dbs” = “1dev1”

  2> go
  ```

  The databases and their devices appear at the destination Adaptive Server, marked as *in-mount*. The system is populated with updates to system catalogs and appropriate information on the databases, but the databases themselves are unrecovered. They will, however, survive a system crash.

  The destination Adaptive Server then recovers the databases one at a time. The databases are left offline after recovery.

  If a recovery fails on a database, it affects only that database. The recovery continues for the other databases.

  Use the command database online to bring the databases online.

  You need not reboot the destination server.

- Using the mount command with listonly displays the path names in the manifest file from the source Adaptive Server without mounting the database.

  Before mounting the databases, use the listonly parameter to list the device path names at the destination Adaptive Server. Then use the mount command to actually mount the databases.

mount database all from <*manifest_file*> with listonly

As an example:

```
1> mount database all from "/data/sybase2/mfile1”
with listonly
2> go

/data/sybase1/d0.dbs = ldev1
```

Once you have the path names, you can verify or modify them to meet your criteria at the destination Adaptive Server.

Renaming devices

The manifest file contains the device paths as known to the source Adaptive Server that created the manifest file. If the destination Adaptive Server accesses the devices with a different path, you can specify the new path to the mount command.

1 Use mount with listonly to display the old path:

```
1> mount database all from “/data/sybase2/mfile1”
with listonly
2> go

“/data/sybase1/d0.dbs” = “1dev1”
```

2 If the new path for the device `“/data/sybase1/d0.dbs”` is `“/sybase/devices/d0.dbs”`, specify the new device in the mount command:

```
mount database all from “/data/sybase2/mfile1” using
“/sybase/devices/d0.dbs” = “ldev1””
```

Destination changes

Once databases are mounted on the destination Adaptive Server, certain settings are cleared on the mounted database:

- Replication is turned off.
- Audit settings are cleared and turned off.
- Omni options, default remote location, and type are cleared.
- Cache bindings are dropped for both the mounted databases and their objects.

- Recovery order is dropped for the mounted databases and becomes the default dbid order.

quiesce database extension

To duplicate or copy databases, use quiesce database with the extension for creating the manifest file. The quiesce database effects the quiesce hold by blocking writes in the database, and then creates the manifest file. The command then returns control of the database to the user.

You can now use a utility to copy the database to another Adaptive Server. These rules for quiesce database hold must be followed for the copy operation:

- The copy operation cannot begin until the quiesce database hold process has completed.
- Every device for every database in the quiesce database command must be copied.
- The copy process must complete before you invoke the quiesce database release.

Syntax

quiesce database < *tag_name* > hold < *dbname list* > [for external dump] [to <*manifest_file*> [with override]]

Parameters

Where:

tag_name
is a user-defined name that designates the list of databases to hold or release. *tag_name* must conform to the rules for identifiers.

dbname list
the list of the databases included in the quiesce database hold command.

for external dump
specifies that while updates to the databases in the list are suspended, you will physically copy all affected database devices, using some facility external to Adaptive Server. The copy operation serves as a replacement for the combination of dump database and load database.

manifest_file
is the binary file that describes the databases included in the command. It can be created only if those databases are isolated and self-contained on their devices.

Examples

Example 1

Place the database in a hold status and build the manifest file using the quiesce command:

```
quiesce database pubs2_tag hold pubs2 for external dump
to "/work2/Devices/Mpubs_file" with override
```

Once the command completes, control returns to the user.

Example 2

Copy the database devices.

If you do not know which devices are to be copied use the mount database with listonly to list all the devices to be copied.

```
1> mount database all from “/data/sybase2/mfile1” with
listonly
2> go

“/data/sybase1/d0.dbs” = “1dev1”
```

A manifest file cannot be created if the set of databases that are quiesced contain references to databases outside of the set. You may use with override option to bypass this restriction.

```
quiesce database pubs2_tag release for external dump to
Mpubs_file
```

CHAPTER 23 Setting Database Options

This chapter describes how to use database options.

What are database options?

Database options control:

- The behavior of transactions
- Defaults for table columns
- Restrictions to user access
- Performance of recovery and bcp operations
- Log behavior

The System Administrator and the Database Owner can use database options to configure the settings for an entire database. Database options differ from sp_configure parameters, which affect the entire server, and set options, which affect only the current session or stored procedure.

Using the *sp_dboption* procedure

Use sp_dboption to change settings for an entire database. The options remain in effect until they are changed. sp_dboption:

- Displays a complete list of the database options when it is used without a parameter
- Changes a database option when used with parameters

You can change options for user databases only. You cannot change options for the master database. To change a database option in a user database (or to display a list of the database options), execute sp_dboption while using the master database.

The syntax is:

sp_dboption [*dbname*, *optname*, {true | false}]

To make an option or options take effect for every new database, change the option in the model database.

Database option descriptions

All users with access to the master database can execute sp_dboption with no parameters to display a list of the database options. The report from sp_dboption looks like this:

```
sp_dboption
Settable database options.
--------------------
abort tran on log full
allow nulls by default
auto identity
dbo use only
ddl in tran
identity in nonunique index
no chkpt on recovery
no free space acctg
read only
select into/bulkcopy/pllsort
single user
trunc log on chkpt
trunc. log on chkpt.
unique auto_identity index
```

For a report on which options have been set in a particular database, execute sp_helpdb in that database.

The following sections describe each database option in detail.

abort tran on log full

abort tran on log full determines the fate of a transaction that is running when the last-chance threshold is crossed. The default value is false, meaning that the transaction is suspended and is awakened only when space has been freed. If you change the setting to true, all user queries that need to write to the transaction log are killed until space in the log has been freed.

allow nulls by default

Setting allow nulls by default to true changes the default null type of a column from not null to null, in compliance with the SQL standard. The Transact-SQL default value for a column is not null, meaning that null values are not allowed in a column unless null is specified in the create table or alter table column definition.

You cannot use allow nulls by default to change the nullibility of a column during select into statements. Instead, use convert() to specify the nullibility of the resulting columns.

auto identity

While the auto identity option is true, a 10-digit IDENTITY column is defined in each new table that is created without specifying either a primary key, a unique constraint, or an IDENTITY column. This IDENTITY column is only created when you issue a create table command, not when you issue a select into. The column is not visible when you select all columns with the select * statement. To retrieve it, you must explicitly mention the column name, SYB_IDENTITY_COL, in the select list.

To set the precision of the automatic IDENTITY column, use the size of auto identity configuration parameter.

Though you can set auto identity to true in tempdb, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column.

dbo use only

While dbo use only is set to true (on), only the Database Owner can use the database.

ddl in tran

Setting ddl in tran to true allows these commands to be used inside a user-defined transaction:

- alter table (clauses other than partition and unpartition are allowed)
- create default
- create index
- create procedure
- create rule
- create schema
- create table
- create trigger
- create view
- drop default
- drop index
- drop procedure
- drop rule
- drop table
- drop trigger
- drop view
- grant
- revoke

Data definition statements lock system tables for the duration of a transaction, which can result in performance problems. Use them only in short transactions.

These commands cannot be used in a user-defined transaction under any circumstances:

- alter database
- alter table...partition
- alter table...unpartition
- create database
- disk init
- dump database
- dump transaction
- drop database
- load transaction
- load database
- select into
- truncate table
- update statistics

identity in nonunique index

identity in nonunique index automatically includes an IDENTITY column in a table's index keys so that all indexes created on the table are unique. This database option makes logically nonunique indexes internally unique and allows those indexes to be used to process updatable cursors and isolation level 0 reads.

The table must already have an IDENTITY column for the identity in nonunique index option to work either from a create table statement or from setting the auto identity database option to true before creating the table.

Use identity in nonunique index if you plan to use cursors and isolation level 0 reads on tables that have nonunique indexes. A unique index ensures that the cursor is positioned at the correct row the next time a fetch is performed on that cursor.

Do not confuse the identity in nonunique index option with unique auto_identity index, which is used to add an IDENTITY column with a unique, nonclustered index to new tables.

no chkpt on recovery

no chkpt on recovery is set to true (on) when an up-to-date copy of a database is kept. In these situations, there is a "primary" database and a "secondary" database. Initially, the primary database is dumped and loaded into the secondary database. Then, at intervals, the transaction log of the primary database is dumped and loaded into the secondary database.

If this option is set to false (off)—the default—a checkpoint record is added to the database after it is recovered by restarting Adaptive Server. This checkpoint, which ensures that the recovery mechanism is not re-run unnecessarily, changes the sequence number of the database. If the sequence number of the secondary database has been changed, a subsequent dump of the transaction log from the primary database cannot be loaded into it.

Turning this option on for the secondary database causes it to not get a checkpoint from the recovery process so that subsequent transaction log dumps from the primary database can be loaded into it.

no free space acctg

no free space acctg suppresses free-space accounting and execution of threshold actions for the non-log segments. This speeds recovery time because the free-space counts will not be recomputed for those segments. It disables updating the rows-per-page value stored for each table, so system procedures that estimate space usage may report inaccurate values.

read only

read only means that users can retrieve data from the database, but cannot modify anything.

select into/bulkcopy/pllsort

select into/bulkcopy/pllsort must be set to on to perform operations that do not keep a complete record of the transaction in the log, which include:

- Using the writetext utility.
- Doing a select into a permanent table.

- Doing a "fast" **bulk copy** with bcp. By default, fast bcp is used on tables that do not have indexes.
- Performing a parallel sort.

Adaptive Server performs minimal logging for these commands, recording only page allocations and deallocations, but not the actual changes made to the data pages.

You do not have to set select into /bulkcopy/pllsort on to select into a user database when you issue the select into command to a temporary table. This is because temporary tables are created on tempdb and tempdb is never recovered. Additionally, you do not need to set the option to run bcp on a table that has indexes, because inserts are logged.

After you have run select into or performed a bulk copy in a database, you will not be able to perform a regular transaction log dump. After you have made minimally logged changes to your database, you must perform a dump database, since changes are not recoverable from transaction logs.

Setting select into/bulkcopy/pllsort does not block log dumping, but making minimally logged changes to data does block the use of a regular dump transaction. However, you can still use dump transaction...with no_log and dump transaction...with truncate_only.

By default, select into/bulkcopy/pllsort is turned off in newly created databases. To change the default, turn this option on in the model database.

single user

When single user is set to true, only one user at a time can access the database. You cannot set single user to true in tempdb.

trunc log on chkpt

When trunc log on chkpt is true (on), the transaction log is truncated (committed transactions are removed) when the checkpoint checking process occurs (usually more than once per minute), if 50 or more rows have been written to the log. The log is *not* truncated if less than 50 rows were written to the log, or if the Database Owner runs the checkpoint command manually.

You may want to turn this option on while doing development work during which backups of the transaction log are not needed. If this option is off (the default), and the transaction log is never dumped, the transaction log continues to grow, and you may run out of space in your database.

When trunc log on chkpt is on, you cannot dump the transaction log because changes to your data are not recoverable from transaction log dumps. Use dump database instead.

By default, the trunc log on chkpt option is off in newly created databases. To change the default, turn this option on in the model database.

Warning! If you set trunc log on chkpt on in model, and you need to load a set of database and transaction logs into a newly created database, be sure to turn the option off in the new database.

unique auto_identity index

When the unique auto_identity index option is set to true, it adds an IDENTITY column with a unique, nonclustered index to new tables. By default, the IDENTITY column is a 10-digit numeric datatype, but you can change this default with the size of auto identity column configuration parameter.

Though you can set unique auto_identity index to true in tempdb, it is not recognized or used, and temporary tables created there do not automatically include an IDENTITY column with a unique index.

The unique auto_identity index option provides a mechanism for creating tables that have an automatic IDENTITY column with a unique index that can be used with updatable cursors. The unique index on the table ensures that the cursor is positioned at the correct row after a fetch. (If you are using isolation level 0 reads and need to make logically nonunique indexes internally unique so that they can process updatable cursors, use the identity in nonunique index option.)

In some cases, the unique auto_identity index option can avoid the Halloween Problem for the following reasons:

- Users cannot update an IDENTITY column; hence, it cannot be used in the cursor update.

- The IDENTITY column is automatically created with a unique, nonclustered index so that it can be used for the updatable cursor scan.

For more information about the Halloween Problem, IDENTITY columns, and cursors, see the *Transact-SQL User's Guide*.

Do not confuse the unique auto_identity index option with the identity in nonunique index option, which is used to make all indexes in a table unique by including an IDENTITY column in the table's index keys.

Changing database options

Only a System Administrator or the Database Owner can change a user's database options by executing sp_dboption. Users aliased to the Database Owner cannot change database options with sp_dboption.

You must be using the master database to execute sp_dboption. Then, for the change to take effect, you must issue the checkpoint command while using the database for which the option was changed.

Remember that you cannot change any master database options.

To change pubs2 to read only:

```
use master
sp_dboption pubs2, "read only", true
```

Then, run the checkpoint command in the database that was changed:

```
checkpoint pubs2
```

For the *optname* parameter of sp_dboption, Adaptive Server understands any unique string that is part of the option name. To set the trunc log on chkpt option:

```
use master
sp_dboption pubs2, trunc, true
```

If you enter an ambiguous value for *optname*, an error message is displayed. For example, two of the database options are dbo use only and read only. Using "only" for the *optname* parameter generates a message because it matches both names. The complete names that match the string supplied are printed out so that you can see how to make the *optname* more specific.

You can turn on more than one database option at a time. You cannot change database options inside a user-defined transaction.

Viewing the options on a database

Use sp_helpdb to determine the options that are set for a particular database. sp_helpdb lists each active option in the "status" column of its output.

The following example shows that the read only option is turned on in mydb:

```
sp_helpdb mydb
```

```
name   db_size  owner  dbid  created         status
-----  -------  -----  ----  ------------  ----------------------
mydb   2.0 MB    sa     5    Mar 05, 1999 read only

device_fragments      size    usage          free kbytes
-----------------     ------  ------------  -------------
master                2.0 MB  data and log           576

device                         segment
------------------------------ ------------------------------
master                         default
master                         logsegment
master                         system

name  attribute_class attribute  int_value char_value                comments
----- --------------- ---------- --------- --------------------     ---------
pubs2 buffer manager  cache name NULL      cache for database mydb  NULL
```

To display a summary of the options for all databases, use sp_helpdb without specifying a database:

```
sp_helpdb
```

```
name           db_size owner dbid created      status
-----------    ------- ----- ---- -----------  ---------------------------
mydb            2.0 MB   sa     5    May 10, 1997  read only
master          3.0 MB   sa     1    Jan 01, 1997  no options set
model           2.0 MB   sa     3    Jan 01, 1997  no options set
```

```
sybsystemprocs 2.0 MB   sa     4    Mar 31, 1995  trunc log on chkpt
tempdb        2.0 MB  sa     2   May 04, 1998 select into/bulkcopy/pllsort
```

CHAPTER 24 Creating and Using Segments

This chapter introduces the system procedures and commands for using segments in databases.

See also Chapter 6, “Controlling Physical Data Placement,” in *Performance and Tuning: Basics* for information about how segments can improve system performance.

What is a segment?

A segment is a label that points to one or more database devices. Segment names are used in create table and create index commands to place tables or indexes on specific database devices. Using segments can improve Adaptive Server performance and give the System Administrator or Database Owner increased control over the placement, size, and space usage of database objects.

You create segments within a database to describe the database devices that are allocated to the database. Each Adaptive Server database can contain up to 32 segments, including the system-defined segments (see "System-defined segments" on page 720). Before assigning segment names, you must initialize the database devices with disk init and then make them available to the database with create database or alter database.

System-defined segments

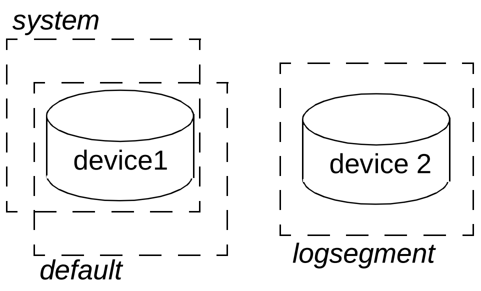
When you first create a database, Adaptive Server creates three segments in the database, as described in Table 24-1.

Table 24-1: System-defined segments

Segment	Function
system	Stores the database's system tables
logsegment	Stores the database's transaction log
default	Stores all other database objects—unless you create additional segments and store tables or indexes on the new segments by using create table...on *segment_name* or create index...on *segment_name*

If you create a database on a single database device, the system, default, and logsegment segments label the same device. If you use the log on clause to place the transaction log on a separate device, the segments resemble those shown in Figure 24-1.

Figure 24-1: System-defined segments



Although you can add and drop user-defined segments, you cannot drop the default, system, or log segments from a database. A database must have at least one default, system-defined, and log segment.

Commands and procedures for managing segments

Table 24-2 summarizes the commands and system procedures for managing segments.

Table 24-2: Commands and procedures for managing segments

Command or procedure	Function
sp_addsegment	Defines a segment in a database.
create table and create index	Creates a database object on a segment.
sp_dropsegment	Removes a segment from a database or removes a single device from the scope of a segment.
sp_extendsegment	Adds devices to an existing segment.
sp_placeobject	Assigns future space allocations for a table or an index to a specific segment.
sp_helpsegment	Displays the segment allocation for a database or data on a particular segment.
sp_helpdb	Displays the segments on each database device. See Chapter 21, "Creating and Managing User Databases," for examples.
sp_help	Displays information about a table, including the segment where the table resides.
sp_helpindex	Displays information about a table's indexes, including the segments where the indexes reside.

Why use segments?

When you add a new device to a database, Adaptive Server places the new device in a default pool of space (the database's default and system segments). This increases the total space available to the database, but it does not determine which objects will occupy that new space. Any table or index might grow to fill the entire pool of space, leaving critical tables with no room for expansion. It is also possible for several heavily used tables and indexes to be placed on a single physical device in the default pool of space, resulting in poor I/O performance.

When you create an object on a segment, the object can use all the database devices that are available in the segment, but no other devices. You can use segments to control the space that is available to individual objects.

The following sections describe how to use segments to control disk space usage and to improve performance. "Moving a table to another device" on page 724 explains how to move a table from one device to another using segments and clustered indexes.

Controlling space usage

If you assign noncritical objects to a segment, those objects cannot grow beyond the space available in the segment's devices. Conversely, if you assign a critical table to a segment, and the segment's devices are not available to other segments, no other objects will compete with that table for space.

When the devices in a segment become full, you can extend the segment to include additional devices or device fragments as needed. Segments also allow you to use thresholds to warn you when space becomes low on a particular database segment.

If you create additional segments for data, you can create new threshold procedures for each segment. See Chapter 31, "Managing Free Space with Thresholds," for more information on thresholds.

Improving performance

In a large, multidatabase and/or multidrive Adaptive Server environment, you can enhance system performance by paying careful attention to the allocation of space to databases and the placement of database objects on physical devices. Ideally, each database has exclusive use of database devices, that is, it does not share a physical disk with another database. In most cases, you can improve performance by placing heavily used database objects on dedicated physical disks or by "splitting" large tables across several physical disks.

The following sections describe these ways to improve performance. The *Performance and Tuning Guide* also offers more information about how segments can improve performance.

Separating tables, indexes, and logs

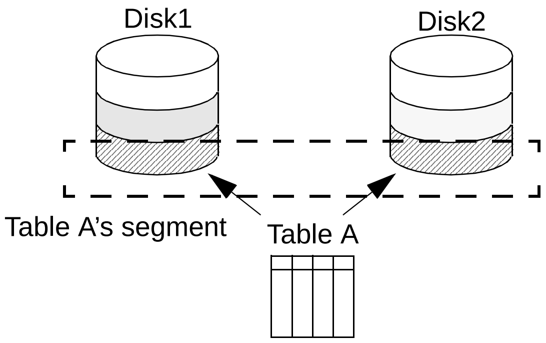
Generally, placing a table on one physical device, its nonclustered indexes on a second physical device, and the transaction log on a third physical device can speed performance. Using separate physical devices (disk controllers) reduces the time required to read or write to the disk. If you cannot devote entire devices in this way, at least restrict all nonclustered indexes to a dedicated physical device.

The log on extension to create database (or sp_logdevice) places the transaction log on a separate physical disk. Use segments to place tables and indexes on specific physical devices. See “Assigning database objects to segments” on page 727 for information about placing tables and indexes on segments.

Splitting tables

You can split a large, heavily used table across devices on separate disk controllers to improve the overall read performance of a table. When a large table exists on multiple devices, it is more likely that small, simultaneous reads will take place on different disks. Figure 24-2 shows a table that is split across the two devices in its segment.

Figure 24-2: Partitioning a table across physical devices



You can split a table across devices using one of three different methods, each of which requires the use of segments:

- Use table partitioning.
- If the table has a clustered index, use partial loading.
- If the table contains text or image datatypes, separate the text chain from other data.

Partitioning tables

Partitioning a table creates multiple page chains for the table and distributes those page chains over all the devices in the table’s segment (see Figure 24-2). Partitioning a table increases both insert and read performance, since multiple page chains are available for insertions.

Before you can partition a table, you must create the table on a segment that contains the desired number of devices. The remainder of this chapter describes how to create and modify segments. See Chapter 6, “Controlling Physical Data Placement,” in *Performance and Tuning: Basics* for information about partitioning tables using alter table.

Partial loading

To split a table with a clustered index, use sp_placeobject with multiple load commands to load different parts of the table onto different segments. This method can be difficult to execute and maintain, but it provides a way to split tables and their clustered indexes across physical devices. See “Placing existing objects on segments” on page 730 for more information and syntax.

Separating text and image columns

Adaptive Server stores the data for text and image columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table’s other data. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain and base table data on a separate physical device can improve performance. See “Placing text pages on a separate device” on page 733 for more information and syntax.

Moving a table to another device

You can also use segments to move a table from one device to another using the create clustered index command. Clustered indexes, where the bottom or *leaf level* of the index contains the actual data, are on the same segment as the table. Therefore, you can completely move a table by dropping its clustered index (if one exists), and creating or re-creating a clustered index on the desired segment. See “Creating clustered indexes on segments” on page 734 for more information and syntax.

Creating segments

To create a segment in a database:

- Initialize the physical device with disk init.
- Make the database device available to the database by using the on clause to create database or alter database. This automatically adds the new device to the database's default and system segments.

Once the database device exists and is available to the database, define the segment in the database with the stored procedure sp_addsegment. The syntax is:

```
sp_addsegment segname, dbname, devname
```

where:

- segname is any valid identifier. Give segments names that identify what they are used for, and use extensions like "_seg."
- dbname is the name of the database where the segment will be created.
- devname is the name of the database device—the name used in disk init and the create and alter database statements.

This statement creates the segment seg_mydisk1 on the database device mydisk1:

```
sp_addsegment seg_mydisk1, mydata, mydisk1
```

Changing the scope of segments

When you use segments, you also need to manage their scope – the number of database devices to which each segment points. You can:

- Extend the scope of a segment by making it point to an additional device or devices, or
- Reduce the scope of a segment by making it point to fewer devices.

Extending the scope of segments

You may need to extend a segment if the database object or objects assigned to the segment run out of space. sp_extendsegment extends the size of a segment by including additional database devices as part of an existing segment. The syntax is:

sp_extendsegment *segname*, *dbname*, *devname*

Before you can extend a segment:

- The database device must be listed in sysdevices,
- The database device must be available in the desired database, and
- The segment name must exist in the current database.

The following example adds the database device pubs_dev2 to an existing segment named bigseg:

```
sp_extendsegment bigseg, pubs2, pubs_dev2
```

To extend the default segment in your database, you must place the word "default" in quotes:

```
sp_extendsegment "default", mydata, newdevice
```

Automatically extending the scope of a segment

If you use alter database to add space on a database device that is new to the database, the system and default segments are extended to include the new space. Thus, the scope of the system and default segments is extended each time you add a new device to the database.

If you use alter database to assign additional space on an existing database device, all the segments mapped to the existing device are extended to include the new device fragment. For example, assume that you initialized a 4MB device named newdev, allocated 2MB of the device to mydata, and assigned the 2MB to the testseg segment:

```
alter database mydata on newdev = "2M"
sp_addsegment testseg, mydata, newdev
```

If you alter mydata later to use the remaining space on newdev, the remaining space fragment is automatically mapped to the testseg segment:

```
alter database mydata on newdev = "2M"
```

See "A segment tutorial" on page 738 for more examples about how Adaptive Server assigns new device fragments to segments.

Reducing the scope of a segment

You may need to reduce the scope of a segment if it includes database devices that you want to reserve exclusively for other segments. For example, if you add a new database device that is to be used exclusively for one table, you will want to reduce the scope of the default and system segments so that they no longer point to the new device.

Use sp_dropsegment to drop a single database device from a segment, reducing the segment's scope:

sp_dropsegment *segname*, *dbname*, *device*

With three arguments, sp_dropsegment drops only the given device from the scope of devices spanned by the segment. You can also use sp_dropsegment to remove an entire segment from the database, as described under "Dropping segments" on page 734.

The following example removes the database device pubs_dev2 from the scope of bigseg:

```
sp_dropsegment bigseg, pubs2, pubs_dev2
```

Assigning database objects to segments

This section explains how to assign new or existing database objects to user-defined segments to:

- Restrict new objects to one or more database devices
- Place a table and its index on separate devices to improve performance
- Split an existing object over multiple database devices

Creating new objects on segments

To place a new object on a segment, first create the new segment. You may also want to change the scope of this segment (or other segments) so that it points only to the desired database devices. Remember that when you add a new database device to a database, it is automatically added to the scope of the default and system segments.

After you have defined the segment in the current database, use create table or create index with the optional on *segment_name* clause to create the object on the segment. The syntax is:

```
create table table_name (col_name datatype ... )
    [on segment_name]
```

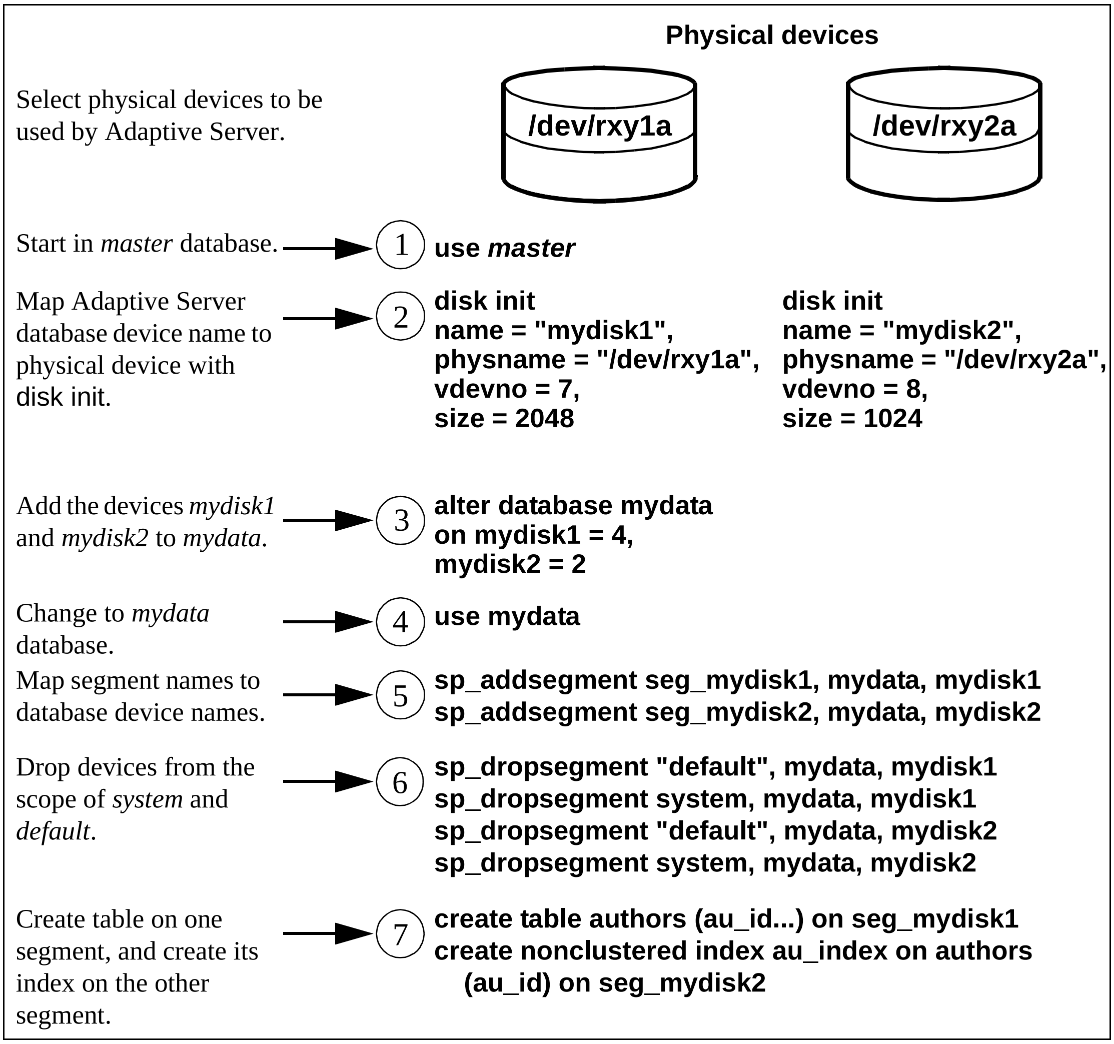
```
create [ clustered | nonclustered ] index index_name
    on table_name(col_name)
    [on segment_name]
```

Note Clustered indexes, where the bottom leaf, or leaf level, of the index contains the actual data, are by definition on the same segment as the table. See "Creating clustered indexes on segments" on page 734.

Example: creating a table and index on separate segments

Figure 24-3 summarizes the sequence of Transact-SQL commands used to create tables and indexes on specific physical disks on a server using 2K logical page size.

Figure 24-3: Creating objects on specific devices using segments


Physical devices
Select physical devices to be used by Adaptive Server.
/dev/rxy1a
/dev/rxy2a
Start in *master* database.
1
use *master*
Map Adaptive Server database device name to physical device with disk init.
2
disk init
name = "mydisk1",
physname = "/dev/rxy1a",
vdevno = 7,
size = 2048
disk init
name = "mydisk2",
physname = "/dev/rxy2a",
vdevno = 8,
size = 1024
Add the devices *mydisk1* and *mydisk2* to *mydata*.
3
alter database mydata
on mydisk1 = 4,
mydisk2 = 2
Change to *mydata* database.
4
use mydata
Map segment names to database device names.
5
sp_addsegment seg_mydisk1, mydata, mydisk1
sp_addsegment seg_mydisk2, mydata, mydisk2
Drop devices from the scope of *system* and *default*.
6
sp_dropsegment "default", mydata, mydisk1
sp_dropsegment system, mydata, mydisk1
sp_dropsegment "default", mydata, mydisk2
sp_dropsegment system, mydata, mydisk2
Create table on one segment, and create its index on the other segment.
7
create table authors (au_id...) on seg_mydisk1
create nonclustered index au_index on authors
(au_id) on seg_mydisk2

1 Start by using the master database.

2 Initialize the physical disks.

3 Allocate the new database devices to a database.

4 Change to the mydata database using the use *database* command.

5 Create two new segments, each of which points to one of the new devices.

6. Reduce the scope of the default and system segments so that they do not point to the new devices.
7. Create the objects, giving the new segment names.

Placing existing objects on segments

sp_placeobject does not remove an object from its allocated segment. However, it causes all further disk allocation for that object to occur on the new segment it specifies. The syntax is:

sp_placeobject *segname*, *objname*

The following command causes all further disk allocation for the mytab table to take place on bigseg:

```
sp_placeobject bigseg, mytab
```

sp_placeobject does not move an object from one database device to another. Whatever pages have been allocated on the first device remain allocated; whatever data was written to the first device remains on the device. sp_placeobject affects only future space allocations.

Note To completely move a table, you can drop its clustered index (if one exists), and create or re-create a clustered index on the desired segment. To completely move a nonclustered index, drop the index and re-create it on the new segment. See “Creating clustered indexes on segments” on page 734 for instructions on moving a table.

After you have used sp_placeobject, executing dbcc checkalloc causes the following message to appear for each object that is split across segments:

```
Extent not within segment: Object object_name, indid
index_id includes extents on allocation page
page_number which is not in segment segment_name.
```

You can ignore this message.

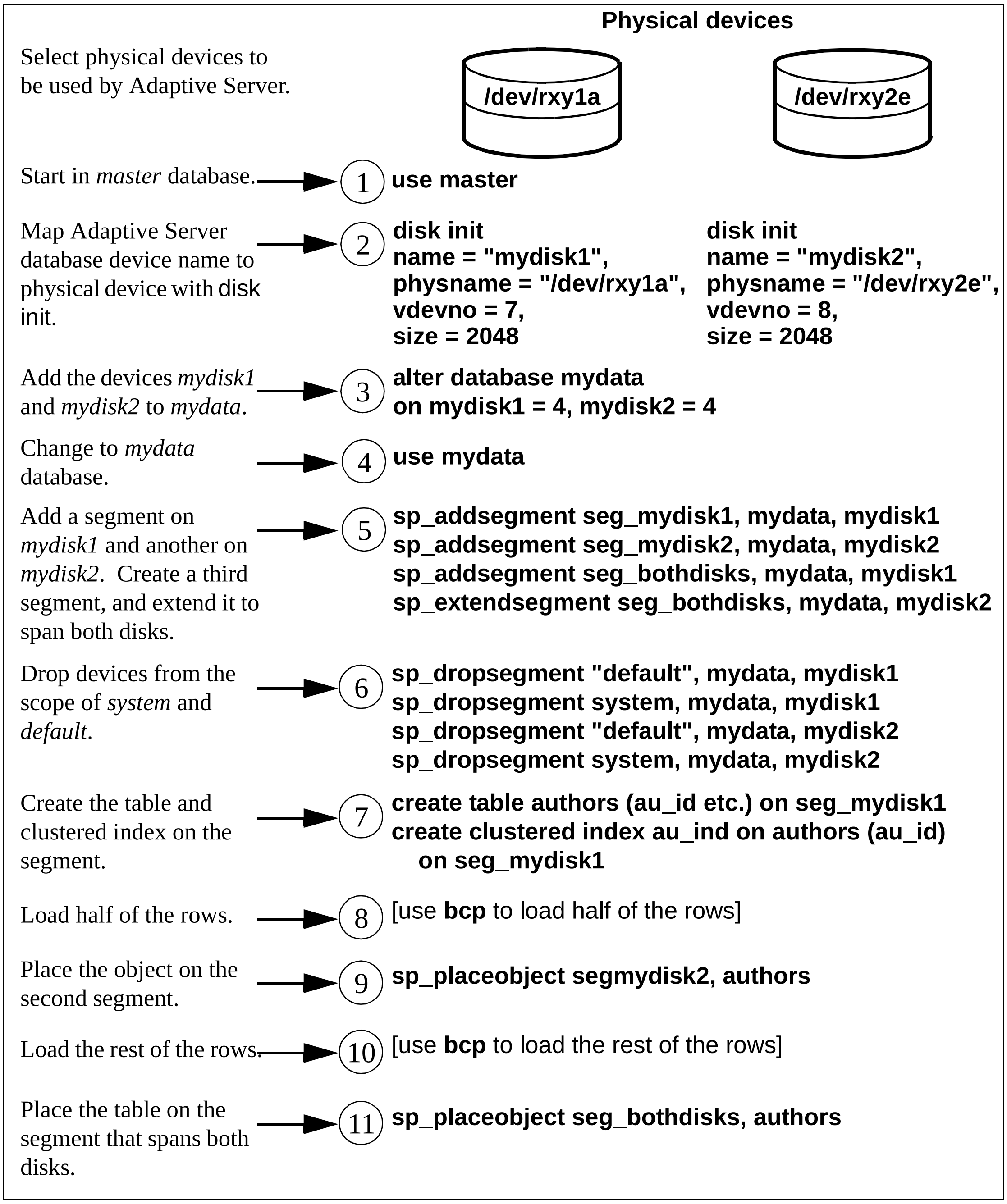
Example: splitting a table and its clustered index across physical devices

Performance can be improved for high-volume, multiuser applications when large tables are split across segments that are located on separate disk controllers.

The order of steps is quite important at certain stages. In particular, you must create the clustered index before you place the table is placed on the second segment.

Figure 24-4 summarizes the process of splitting a table across two segments on a server using 2K logical page size:

Figure 24-4: Splitting a large table across two segments



Select physical devices to be used by Adaptive Server.

Start in *master* database. → 1
```
use master
```

Map Adaptive Server database device name to physical device with disk init. → 2
```
disk init                       disk init
name = "mydisk1",               name = "mydisk2",
physname = "/dev/rxy1a",        physname = "/dev/rxy2e",
vdevno = 7,                     vdevno = 8,
size = 2048                     size = 2048
```

Add the devices *mydisk1* and *mydisk2* to *mydata*. → 3
```
alter database mydata
on mydisk1 = 4, mydisk2 = 4
```

Change to *mydata* database. → 4
```
use mydata
```

Add a segment on *mydisk1* and another on *mydisk2*. Create a third segment, and extend it to span both disks. → 5
```
sp_addsegment seg_mydisk1, mydata, mydisk1
sp_addsegment seg_mydisk2, mydata, mydisk2
sp_addsegment seg_bothdisks, mydata, mydisk1
sp_extendsegment seg_bothdisks, mydata, mydisk2
```

Drop devices from the scope of *system* and *default*. → 6
```
sp_dropsegment "default", mydata, mydisk1
sp_dropsegment system, mydata, mydisk1
sp_dropsegment "default", mydata, mydisk2
sp_dropsegment system, mydata, mydisk2
```

Create the table and clustered index on the segment. → 7
```
create table authors (au_id etc.) on seg_mydisk1
create clustered index au_ind on authors (au_id)
   on seg_mydisk1
```

Load half of the rows. → 8 [use **bcp** to load half of the rows]

Place the object on the second segment. → 9
```
sp_placeobject segmydisk2, authors
```

Load the rest of the rows. → 10 [use **bcp** to load the rest of the rows]

Place the table on the segment that spans both disks. → 11
```
sp_placeobject seg_bothdisks, authors
```

1. Begin by using the master database.
2. Initialize the devices with disk init.
3. Assign both devices to the mydata database with alter database.
4. Change to the mydata database by entering the use *database* command.
5. Create three segments. The first two should each point to one of the new devices. Extend the scope of the third segment so that it labels both devices.
6. Drop the system and default segments from both devices.
7. Create the table and its clustered index on the first segment.
8. Load half of the table's data onto the first segment.
9. Use sp_placeobject to cause all further allocations of disk space to occur on the second segment.
10. Load the remaining data onto the second segment.
11. Use sp_placeobject again to place the table on the segment that spans both devices.

The balance of disk allocation may change over time if the table is updated frequently. To guarantee that the speed advantages are maintained, you may need to drop and re-create the table at some point.

Placing text pages on a separate device

When you create a table with text or image columns, the data is stored on a separate chain of text pages. A table with text or image columns has an additional entry in sysindexes for the text chain, with the name column set to the name of the table preceded by the letter "t" and an indid of 255. You can use sp_placeobject to store the text chain on a separate device, giving both the table name and the name of the text chain from sysindexes:

```
sp_placeobject textseg, "mytab.tmytab"
```

Note By default, a chain of text pages is placed on the same segment as its table. After you execute sp_placeobject, pages that were previously written on the old device remain allocated, but all new allocations take place on the new segment.

If you want the text pages to be on a particular segment, first create the table on that segment (allocating the initial extent on that segment), then create a clustered index on the table to move the rest of the data to the segment.

Creating clustered indexes on segments

The bottom, or leaf level, of a clustered index contains the data. Therefore, a table and its clustered index are on the same segment. If you create a table on one segment and its clustered index on a different segment, the table will migrate to the segment where you created the clustered index. This provides a quick and easy way to move a table to other devices in your database.

The syntax for creating a clustered index on a segment is:

```
create [unique] clustered index index_name
    on [[database.]owner.]table_name (column_name
        [, column_name]...)
    [with {fillfactor = x, ignore_dup_key, sorted_data,
        [ignore_dup_row | allow_dup_row]}]
    on segment_name
```

See “Segments and clustered indexes” on page 743 for an example of this command.

Dropping segments

When you use sp_dropsegment with only a segment name and the database name, the named segment is dropped from the database. However, you cannot drop a segment as long as database objects are still assigned to it. You must assign the objects to another segment or drop the objects first and then drop the segment.

The syntax for dropping a segment is:

sp_dropsegment *segname*, *dbname*

You cannot completely drop the default, system, or log segment from a database. A database must have at least one default, system, and log segment. You can, however, reduce the scope of these segments–see "Reducing the scope of a segment" on page 727.

Note Dropping a segment removes its name from the list of segments in the database, but it does not remove database devices from the allocation for that database, nor does it remove objects from devices.
If you drop all segments from a database device, the space is still allocated to the database but cannot be used for database objects. dbcc checkcatalog reports "Missing segment in Sysusages segmap." To make a device available to a database, use sp_extendsegment to map the device to the database's default segment:

```
sp_extendsegment "default", dbname, devname
```

Getting information about segments

Four system procedures provide information about segments:

- sp_helpsegment lists the segments in a database or displays information about a particular segment in the database.
- sp_helpdb displays information about the relationship between devices and segments in a database.
- sp_help and sp_helpindex display information about tables and indexes, including the segment to which the object is assigned.

sp_helpsegment

sp_helpsegment, when used without an argument, displays information about all of the segments in the database where you execute it:

```
sp_helpsegment
segment name                              status
------- ----------------------------- ------
```

```
      0 system                          0
      1 default                         1
      2 logsegment                      0
      3 seg1                            0
      4 seg2                            0
```

For information about a particular segment, specify the segment name as an argument. Use quotes when requesting information about the default segment:

```
sp_helpsegment "default"
```

The following example displays information about seg1:

```
sp_helpsegment seg1

segment name                           status
------- ------------------------------ ------
      4 seg1                                0

device                 size             free_pages
---------------------- ---------------- -----------
user_data10            15.0MB                  6440
user_data11            15.0MB                  6440
user_data12            15.0MB                  6440

table_name             index_name            indid
---------------------- --------------------- ------
customer               customer                   0

total_size      total_pages free_pages  used_pages
--------------- ----------- ----------- -----------
45.0MB                23040       19320        3720
```

sp_helpdb

When you execute sp_helpdb within a database, and give that database's name, you see information about the segments in the database.

For example:

```
sp_helpdb mydata
```

```
name      db_size    owner  dbid created        status
--------- ---------- ----------- -------------- ------------
mydata        8.0 MB sa        4 May 27, 1993   no options set
```

```
device_fragments    size       usage            free kbytes
------------------- ---------- ---------------- -----------
datadev2            4.0 MB     data only               3408
logdev              2.0 MB     log only                2032
seg_mydisk1         2.0 MB     data only               2016

device                         segment
------------------------------ ------------------------
datadev2                       default
datadev2                       system
logdev                         logsegment
seg_mydisk1                    seg1
```

sp_help and *sp_helpindex*

When you execute sp_help and sp_helpindex in a database, and give a table's name, you see information about which segment(s) stores the table or its indexes.

For example:

```
    sp_helpindex authors
```

```
index_name     index_description                       index_keys
------------   -----------------------------------     ----------
au_index       nonclustered located on seg_mydisk2     au_id
```

Segments and system tables

Three system tables store information about segments: master..sysusages and two system tables in the user database, sysindexes and syssegments. sp_helpsegment uses these tables. Additionally, it finds the database device name in sysdevices.

When you allocate a device to a database with create database or alter database, Adaptive Server adds a row to master..sysusages. The segmap column in sysusages provides bitmaps to the segments in the database for each device.

create database also creates the syssegments table in the user database with these default entries:

```
segment name            status
------- --------------- ------
      0 system               0
      1 default              1
      2 logsegment           0
```

When you add a segment to a database with sp_addsegment, the procedure:

- Adds a new row to the syssegments table in the user database, and
- Updates the segmap in master..sysusages.

When you create a table or an index, Adaptive Server adds a new row to sysindexes. The segment column in that table stores the segment number, showing where the server will allocate new space for the object. If you do not specify a segment name when you create the object, it is placed on the default segment; otherwise, it is placed on the specified segment.

If you create a table containing text or image columns, a second row is also added to sysindexes for the linked list of text pages; by default, the chain of text pages is stored on the same segment as the table. An example using sp_placeobject to put the text chain on its own segment is included under "A segment tutorial" on page 738.

The name from syssegments is used in create table and create index statements. The status column indicates which segment is the default segment.

Note See "System tables that manage space allocation" on page 688 for more information about the segmap column and the system tables that manage storage.

A segment tutorial

The following tutorial shows how to create a user segment and how to remove all other segment mappings from the device. The examples in this section assume a server using 2K logical page sizes.

When you are working with segments and devices, remember that:

- If you assign space in fragments, each fragment will have an entry in sysusages.
- When you assign an additional fragment of a device to a database, all segments mapped to the existing fragment are mapped to the new fragment.
- If you use alter database to add space on a device that is new to the database, the system and default segments are automatically mapped to the new space.

The tutorial begins with a new database, created with one device for the database objects and another for the transaction log:

```
create database mydata on bigdevice = "4M"
     log on logdev = "2M"
```

Now, if you use mydata, and run sp_helpdb, you see:

```
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       6.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage          free kbytes
---------------------- ------------- -------------- -----------
bigdevice              4.0 MB        data only             3408
logdev                 2.0 MB        log only              2032

device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
```

Like all newly created databases, mydata has the segments named default, system, and logsegment. Because create database used log on, the logsegment is mapped to its own device, logdev, and the default and system segments are both mapped to bigdevice.

If you add space on the same database devices to mydata, and run sp_helpdb again, you see entries for the added fragments:

```
use master
alter database mydata on bigdevice = "2M"
    log on logdev = "1M"
```

```
                              use mydata
                              sp_helpdb mydata
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata       9.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage           free kbytes
---------------------- ------------- --------------- -----------
bigdevice              2.0 MB        data only              2048
bigdevice              4.0 MB        data only              3408
logdev                 1.0 MB        log only               1024
logdev                 2.0 MB        log only               2032

device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
```

Always add log space to log space and data space to data space. Adaptive Server instructs you to use with override if you try to allocate a segment that is already in use for data to the log, or vice versa. Remember that segments are mapped to entire devices, and not just to the space fragments. If you change any of the segment assignments on a device, you make the change for all of the fragments.

The following example allocates a new database device that has not been used by mydata:

```
                              use master
                              alter database mydata on newdevice = 3
                              use mydata
                              sp_helpdb mydata
name       db_size  owner     dbid   created      status
---------- -------- --------- ------ ------------ ---------------
mydata      12.0 MB sa             4 May 27, 1993  no options set

device_fragments       size          usage           free kbytes
---------------------- ------------- --------------- -----------
bigdevice              2.0 MB        data only              2048
bigdevice              4.0 MB        data only              3408
logdev                 1.0 MB        log only               1024
logdev                 2.0 MB        log only               2032
newdevice              3.0 MB        data only              3072
```

```
device                 segment
---------------------- ----------------------
bigdevice              default
bigdevice              system
logdev                 logsegment
newdevice              default
newdevice              system
```

The following example creates a segment called new_space on newdevice:

```
sp_addsegment new_space, mydata, newdevice
```

Here is the portion of the sp_helpdb report which lists the segment mapping:

```
device                          segment
----------------------------    ------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       default
newdevice                       new_space
newdevice                       system
```

The default and system segments are still mapped to newdevice. If you are planning to use new_space to store a user table or index for improved performance, and you want to ensure that other user objects are not stored on the device by default, reduce the scope of default and system with sp_dropsegment:

```
sp_dropsegment system, mydata, newdevice
sp_dropsegment "default", mydata, newdevice
```

You must include the quotes around "default" it is a Transact-SQL reserved word.

Here is the portion of the sp_helpdb report that shows the segment mapping:

```
device                          segment
----------------------------    --------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       new_space
```

Only new_space is now mapped to newdevice. Users who create objects can use on new_space to place a table or index on the device that corresponds to that segment. Since the default segment is not pointing to that database device, users who create tables and indexes without using the on clause will not be placing them on your specially prepared device.

If you use alter database on newdevice again, the new space fragment acquires the same segment mapping as the existing fragment of that device (that is, the new_space segment only).

At this point, if you use create table and name new_space as the segment, you will get results like these from sp_help and sp_helpsegment:

```
create table mytabl (c1 int, c2 datetime)
    on new_space
sp_help mytabl

Name              Owner             Type
----------------- ----------------- ----------------
mytabl            dbo               user table

Data_located_on_segment        When_created
------------------------------ --------------------
new_space                      May 27 1993  3:21PM

Column_name   Type      Length Nulls Default_name Rule_name
------------- --------- ------ ----- ------------ ----------
c1            int            4     0 NULL           NULL
c2            datetime       8     0 NULL           NULL
Object does not have any indexes.
No defined keys for this object.

sp_helpsegment new_space

segment name                           status
------- ------------------------------ ------
      3 new_space                           0

device                 size          free_pages
---------------------- ------------- -----------
newdevice              3.0MB                1528

table_name            index_name             indid
--------------------- ---------------------- ------
mytabl                mytabl                      0

total_size       total_pages free_pages  used_pages
```

```
--------------- ----------- ----------- -----------
3.0MB                  1536        1528           8
```

Segments and clustered indexes

This example creates a clustered index, without specifying the segment name, using the same table you just created on the new_space segment in the preceding example. Check new_space after create index to verify that no objects remain on the segment:

```
/* Don't try this at home */
create clustered index mytabl_cix
    on mytabl(c1)
sp_helpsegment new_space
segment name                           status
------- ------------------------------ ------
      3 new_space                           0

device                 size          free_pages
---------------------- ------------- -----------
newdevice              3.0MB                1528

total_size       total_pages free_pages  used_pages
--------------- ----------- ----------- -----------
3.0MB                  1536        1528           8
```

If you have placed a table on a segment, and you need to create a clustered index, use the on *segment_name* clause, or the table will migrate to the default segment.

CHAPTER 25 Using the *reorg* Command

Update activity against a table can eventually lead to inefficient utilization of space and reduced performance. The reorg command reorganizes the use of table space and improves performance.

reorg subcommands

The reorg command provides four subcommands for carrying out different types and levels of reorganization:

- reorg forwarded_rows undoes row forwarding.
- reorg reclaim_space reclaims unused space left on a page as a result of deletions and row-shortening updates.
- reorg compact both reclaims space and undoes row forwarding.
- reorg rebuild undoes row forwarding and reclaims unused page space, as does reorg compact. In addition, reorg rebuild:
    - Rewrites all rows to accord with a table's clustered index, if it has one
    - Writes rows to data pages to accord with any changes made in space management settings through sp_chgattribute

- Drops and re-creates all indexes belonging to the table

The reclaim_space, forwarded_rows, and compact subcommands:

- Minimize interference with other activities by using multiple small transactions of brief duration. Each transaction is limited to eight pages of reorg processing.
- Provide resume and time options that allow you to set a time limit on how long a reorg runs and to resume a reorg from the point at which the previous reorg stopped. This allows you to, for example, use a series of partial reorganizations at off-peak times to reorg a large table. For more information, see "resume and time options for reorganizing large tables" on page 756.

The following considerations apply to the rebuild subcommand:

- reorg rebuild holds an exclusive table lock for its entire duration. On a large table this may be a significant amount of time. However, reorg rebuild accomplishes everything that dropping and re-creating a clustered index does and takes less time. In addition, reorg rebuild rebuilds the table using all of the table's current space management settings. Dropping and re-creating an index does not use the space management setting for reservepagegap.
- In most cases, reorg rebuild requires additional disk space equal to the size of the table it is rebuilding and its indexes.

The following restrictions hold:

- The table specified in the command, if any, must use either the datarows or datapages locking scheme.
- You must be a System Administrator or the object owner to issue reorg.
- You cannot issue reorg within a transaction.

When to run a *reorg* command

reorg is useful when:

- A large number of forwarded rows causes extra I/O during read operations.

- Inserts and serializable reads are slow because they encounter pages with noncontiguous free space that needs to be reclaimed.
- Large I/O operations are slow because of low cluster ratios for data and index pages.
- sp_chgattribute was used to change a space management setting (reservepagegap, fillfactor, or exp_row_size) and the change is to be applied to all existing rows and pages in a table, not just to future updates.

Using the *optdiag* utility to assess the need for a *reorg*

To assess the need for running a reorg, you can use statistics from the systabstats table and the optdiag utility. systabstats contains statistics on the utilization of table space, while optdiag generates reports based on statistics in both systabstats and the sysstatistics table.

For information on the systabstats table, see the *Performance and Tuning Guide*. For information about optdiag, see *Utility Programs for UNIX Platforms*.

Space reclamation without the *reorg* command

Several types of activities reclaim or reorganize the use of space in a table on a page-by-page basis:

- Inserts, when an insert encounters a page that would have enough room if it reclaimed unused space.
- The update statistics command (for index pages only)
- Re-creating clustered indexes
- The housekeeper garbage collection task, if enable housekeeper GC is set to 1 or more

Each of these has limitations and may be insufficient for use on a large number of pages. For example, inserts may execute more slowly when they need to reclaim space, and may not affect many pages with space that can be reorganized. Space reclamation under the housekeeper garbage collection task compacts unused space, but a single housekeeper garbage collection task that runs at user priority may not reach every page that needs it.

Moving forwarded rows to home pages

If an update makes a row too long to fit on its current page, the row is forwarded to another page. A reference to the row is maintained on its original page, the row's *home* page, and all access to the forwarded row goes through this reference. Thus, it always takes two page accesses to get to a forwarded row. If a scan needs to read a large number of forwarded pages, the I/Os caused by extra page accesses slow performance.

reorg forwarded_rows undoes row forwarding by either moving a forwarded row back to its home page, if there is enough space, or by deleting the row and reinserting it in a new home page.

You can get statistics on the number of forwarded rows in a table by querying systabstats and using optdiag.

reorg forwarded_rows syntax

The syntax for reorg forwarded_rows is:

```
reorg forwarded_rows tablename
    [with {resume, time = no_of_minutes}]
```

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 756.

reorg forwarded_rows does not apply to indexes, because indexes do not have forwarded rows.

Using *reorg compact* to remove row forwarding

reorg forwarded_rows uses allocation page hints to find forwarded rows. Because it does not have to search an entire table, this command executes quickly, but it may miss some forwarded rows. After running reorg forwarded_rows, you can evaluate its effectiveness by using optdiag and checking "Forwarded row count." If "Forwarded row count" is high, you can then run reorg compact, which goes through a table page by page and undoes all row forwarding.

Reclaiming unused space from deletes and updates

When a task performs a delete operation or an update that shortens row length, the empty space is preserved in case the transaction is rolled back. If a table is subject to frequent deletes and row-shortening updates, unreclaimed space may accumulate to the point that it impairs performance.

reorg reclaim_space reclaims unused space left by deletes and updates. On each page that has space resulting from committed deletes or row-shortening updates, reorg reclaim_space rewrites the remaining rows contiguously, leaving all the unused space at the end of the page. If all rows have been deleted and there are no remaining rows, reorg reclaim_space deallocates the page.

You can get statistics on the number of unreclaimed row deletions in a table from the systabstats table and by using the optdiag utility. There is no direct measure of how much unused space there is as a result of row-shortening updates.

reorg reclaim_space syntax

The syntax for reorg reclaim_space is:

```
reorg reclaim_space tablename [indexname]
    [with {resume, time = no_of_minutes}]
```

If you specify only a table name, only the table's data pages are reorganized to reclaim unused space; in other words, indexes are not affected. If you specify an index name, only the pages of the index are reorganized.

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 756.

Reclaiming unused space and undoing row forwarding

reorg compact combines the functions of reorg reclaim_space and reorg forwarded_rows. Use reorg compact when:

- You do not need to rebuild an entire table (reorg rebuild); however, both row forwarding and unused space from deletes and updates may be affecting performance.
- There are a large number of forwarded rows. See "Using reorg compact to remove row forwarding" on page 749.

reorg compact syntax

The syntax for reorg compact is:

```
reorg compact tablename
    [with {resume, time = no_of_minutes}]
```

For information about the resume and time options, see "resume and time options for reorganizing large tables" on page 756.

Rebuilding a table

Use reorg rebuild when:

- Large I/O is not being selected for queries where it is usually used, and optdiag shows a low cluster ratio for datapages, data rows, or index pages.
- You used sp_chgattribute to change one or more of the exp_row_size, reservepagegap, or fillfactor space management settings and you want the changes to apply not only to future data, but also to existing rows and pages. For information about sp_chgattribute, see the *Reference Manual*.

If a table needs to be rebuilt because of a low cluster ratio, it may also need to have its space management settings changed (see "Changing space management settings before using reorg rebuild" on page 752).

reorg rebuild uses a table's current space management settings to rewrite the rows in the table according to the table's clustered index, if it has one. All indexes on the table are dropped and re-created using the current space management values for reservepagegap and fillfactor. After a rebuild, a table has no forwarded rows and no unused space from deletions or updates.

reorg rebuild syntax

The syntax for reorg rebuild is:

reorg rebuild *tablename* [*index_name*]

reorg rebuild performs the following when you run it against a table:

- Takes an exclusive table lock
- Copies data from old to new pages
- Deallocates old data pages
- Locks system tables for updates (including sysindexes, sysobjects, syspartitions, and systabstats
- Rebuilds clustered and non-clusterd indexes against new data pages
- Commits all open transactions
- Releases locks on system tables.

If the table is large and has several indexes, the locks for updating system tables can be held for a long time, and these locks may block other processes from accessing these system tables. However, because systabstats is already datarow locked, this system table does not impact this blocking

reorg rebuild builds the clustered index using the with sorted data option, so the data does not have to be resorted during this index build.

Prerequisites for running *reorg rebuild*

Before you run reorg rebuild on a table:

- Set the database option select into/bulkcopy/pllsort to true and run checkpoint in the database.
- Make sure that additional disk space, equal to the size of the table and its indexes, is available.

To set select into/bulkcopy/pllsort to true and checkpoint the database, use the following isql commands:

```
1> use master
2> go
1> sp_dboption pubs2,
    "select into/bulkcopy/pllsort", true
2> go
1> checkpoint pubs2
```

```
2> go
```

Following a rebuild on a table:

- You must dump the database containing the table before you can dump the transaction log.
- Distribution statistics for the table are updated.
- All stored procedures that reference the table will be recompiled the next time they are run.

Changing space management settings before using *reorg rebuild*

When reorg rebuild rebuilds a table, it rewrites all table and index rows according to the table's current settings for reservepagegap, fillfactor, and exp_row_size. These properties all affect how quickly inserts cause a table to become fragmented, as measured by a low cluster ratio.

If it appears that a table quickly becomes fragmented and needs to be rebuilt too frequently, it may be a sign that you need to change the table's space management settings before you run reorg rebuild.

To change the space management settings, use sp_chgattribute (see the *Reference Manual*). For information on space management settings, see the *Performance and Tuning Guide*.

Using the *reorg rebuild* command on indexes

The reorg rebuild command allows you to rebuild individual indexes while the table itself is accessible for read and update activities.

Syntax

The syntax for rebuilding an index is:

reorg rebuild *table_name index_name*

Comments

To use reorg rebuild, you must be the table owner or the Database Owner, or have System Administrator privileges.

If you omit the index name, the entire table is rebuilt.

If you specify an index, only that index is rebuilt.

Requirements for using reorg rebuild on an index are less stringent than for tables. The following rules apply:

- You do not need to set select into to rebuild an index.
- Rebuilding a table requires space for a complete copy of the table. Rebuilding an index works in small transactions, and deallocates pages once they are copied; therefore, the process only needs space for the pages copied on each transaction.
- You can rebuild the index on a table while transaction level scans (dirty reads) are active.

Limitations

The reorg command applies only to tables using datarows or datapages locking. You cannot run reorg on a table that uses allpages locking.

You cannot run reorg on a text index, the name from sysindexes associated with a text chain.

You cannot run reorg within a transaction.

You can do a dump tran on a table after rebuilding its index. However, you cannot do a dump tran if the entire table has been rebuilt.

You can rebuild the index for systabstats, but you cannot run reorg rebuild on the table itself.

Although online index rebuilding is allowed on a placement index, it rebuilds only the index pages. The data pages remain untouched, which means datarows are neither sorted nor rewritten to fresh pages. You can rebuild data pages by dropping a placement index, and then re-creating it.

How indexes are rebuilt with *reorg rebuild indexname*

Rebuilding a single index rewrites all index rows to new pages. This improves performance by:

- Improving clustering of the leaf level of the index
- Applying stored values for the fill factor on the index, which can reduce page splits
- Applying any stored value for reservepagegap, which can help reserve pages for future splits

To reduce contention with users whose queries need to use the index, reorg rebuild locks a small number of pages at a time. Rebuilding an index is a series of independent transactions, with some independent, nested transactions. Approximately 32 pages are rebuilt in each nested transaction and approximately 256 pages are rebuilt in each outer transaction. Address locks are acquired on the pages being modified and are released at the end of the topaction. The pages deallocated in a transaction are not available for reuse until the next transaction begins.

If the reorg rebuild command stops running, the transactions that are already committed are not rolled back. Therefore, the part that has been reorganized is well clustered with desired space utilization, and the part that has not been reorganized is the same as it was before you ran the command. The index remains logically consistent.

Note Rebuilding the clustered index does not affect the data pages of the table. It only affects the leaf pages and higher index levels. Non-leaf pages above level 1 are not rebuilt.

Space requirements for rebuilding an index

If you do not specify fill_factor or reservepagegap, rebuilding an index requires additional space of approximately 256 pages or less in the data segment. The amount of log space required is larger than that required to drop the index and re-create it using create index, but it should be only a small fraction of the actual index size. The more additional free space is available, the better the index clustering will be.

Note reorg rebuild may not rebuild those parts of the index that are already well clustered and have the desired space utilization.

Performance characteristics

Index scans are faster after you run reorg.

Running reorg against a table can have a negative effect on performance of concurrent queries.

Status messages

Running reorg rebuild *indexname* on a large table may take a long time. Periodic status messages are printed to give the user an idea of how reorg has progressed. Starting and ending messages are written to the error log and to the client process executing reorg. In-progress messages go only to the client.

A status reporting interval is calculated as either 10 percent of the pages to be processed or 10,000 pages, whichever is larger. When this number of pages is processed, a status message is printed. Therefore, no more than 10 messages are printed, regardless of the size of the index. Status messages for existing reorg commands are printed more frequently.

resume and *time* options for reorganizing large tables

Use the resume and time options of the reorg command when reorganizing an entire table would take too long and interfere with other database activities. time allows you to run a reorg for a specified length of time. resume allows you to start a reorg at the point in a table where the previous reorg left off. In combination, the two options allow you to reorganize a large table by running a series of partial reorganizations (for example, during off-hours).

resume and time not available with reorg rebuild.

Syntax for using *resume* and *time* in *reorg* commands

The syntax for resume and time is:

```
reorg reclaim_space tablename [indexname]
    [with {resume, time = no_of_minutes}]

reorg forwarded_rows tablename
    [with {resume, time = no_of_minutes}]

reorg compact tablename
    [with {resume, time = no_of_minutes}]
```

The following considerations apply:

- If you specify only the resume option, the reorg begins at the point where the previous reorg stopped and continues to the end of the table.
- If you specify only the time option, the reorg starts at the beginning of the table and continues for the specified number of minutes.
- If you specify both options, the reorg starts at the point where the previous reorg stopped and continues for the specified number of minutes.

Specifying *no_of_minutes* in the *time* option

The *no_of_minutes* argument in the time option refers to elapsed time, not CPU time. For example, to run reorg compact for 30 minutes, beginning where a previous reorg compact finished, enter:

```
reorg compact tablename with resume, time=30
```

If the reorg process goes to sleep during any part of the 30 minutes, it still counts as part of the elapsed time and does not add to the duration of the reorg.

When the amount of time specified has passed, reorg saves statistics about the portion of the table or index that was processed in the systabstats table. This information is used as the restart point for a reorg with the resume option. The restart points for each of the three subcommands that take resume and time options are maintained separately. You cannot, for example, start a reorg with reorg reclaim_space and then resume it with reorg compact.

If you specify *no_of_minutes*, and reorg arrives at the end of a table or an index before the time is up, it returns to the beginning of the object and continues until it reaches its time limit.

Note resume and time allow you to reorganize an entire table or index over multiple runs. However, if there are updates between reorg runs, some pages may be processed twice and some pages may not be processed at all.

CHAPTER 26 Checking Database Consistency

This chapter describes how to check database consistency and perform some kinds of database maintenance using the dbcc commands.

What is the database consistency checker?

The database consistency checker (dbcc) provides commands for checking the logical and physical consistency of a database. Two major functions of dbcc are:

- Checking page linkage and data pointers at both the page level and the row level using checkstorage or checktable and checkdb
- Checking page allocation using checkstorage, checkalloc, checkverify, tablealloc, and indexalloc

dbcc checkstorage stores the results of checks in the dbccdb database. You can print reports from dbccdb using the dbcc stored procedures.

Use the dbcc commands:

- As part of regular database maintenance – the integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database for additional confidence in the integrity of the backup.
- If you suspect that a database is damaged – for example, if using a particular table generates the message "Table corrupt," you can use dbcc to determine if other tables in the database are also damaged.

If you are using Component Integration Services, there are additional dbcc commands you can use for remote databases. For more information, see the *Component Integration Services User's Guide*.

Understanding page and object allocation concepts

When you initialize a database device, the disk init command divides the new space into **allocation units**. The size of the allocation unit depends on the size of the logical pages your server uses (2, 4, 8, or 16K). The first page of each allocation unit is an **allocation page**, which tracks the use of all pages in the allocation unit. Allocation pages have an object ID of 99; they are not real database objects and do not appear in system tables, but dbcc errors on allocation pages report this value.

When a table or an index requires space, Adaptive Server allocates a block of 8 pages to the object. This 8-page block is called an *extent*. Each allocation unit contains 32 extents. The size of the extent also depends on the size of the server logical pages. Adaptive Server uses extents as a unit of space management to allocate and deallocate space as follows:

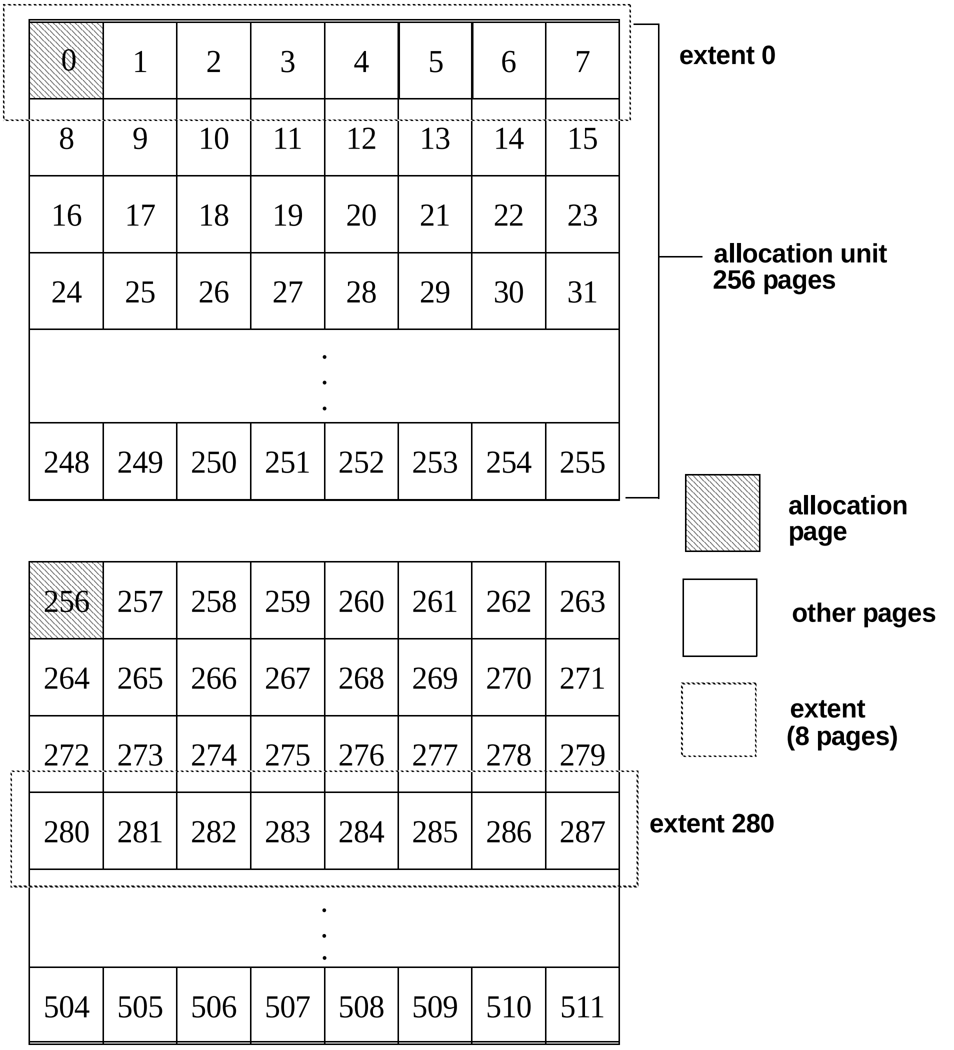
- When you create a table or an index, Adaptive Server allocates an extent for the object.

- When you add rows to an existing table, and the existing pages are full, Adaptive Server allocates another page. If all pages in an extent are full, Adaptive Server allocates an additional extent.
- When you drop a table or an index, Adaptive Server deallocates the extents it occupied.
- When you delete rows from a table so that it shrinks by a page, Adaptive Server deallocates the page. If the table shrinks off the extent, Adaptive Server deallocates the extent.

Every time space is allocated or deallocated on an extent, Adaptive Server records the event on the allocation page that tracks the extents for that object. This provides a fast method for tracking space allocations in the database, since objects can shrink or grow without excess overhead.

Figure 26-1 shows how data pages are set up within extents and allocation units in Adaptive Server databases.

Figure 26-1: Page management with extents



dbcc checkalloc checks all allocation pages (page 0 and all pages divisible by 256) in a database and reports on the information it finds. dbcc indexalloc and dbcc tablealloc check allocation for specific database objects.

Understanding the object allocation map (OAM)

Each table and index on a table has an *Object Allocation Map* (*OAM*). The OAM is stored on pages allocated to the table or index and is checked when a new page is needed for the index or table. A single OAM page can hold allocation mapping for between 2,000 and 63,750 data or index pages. Each OAM page is the size of one logical page size. For example, on a server using a logical page size of 4K, each OAM page is 4K.

The number of entries per OAM page also depends on the logical page size the server is using. The following table describes the number of OAM entries for each logical page size:

2K logical page size	4K logical page size	8K logical page size	16K logical page size
250	506	1018	2042

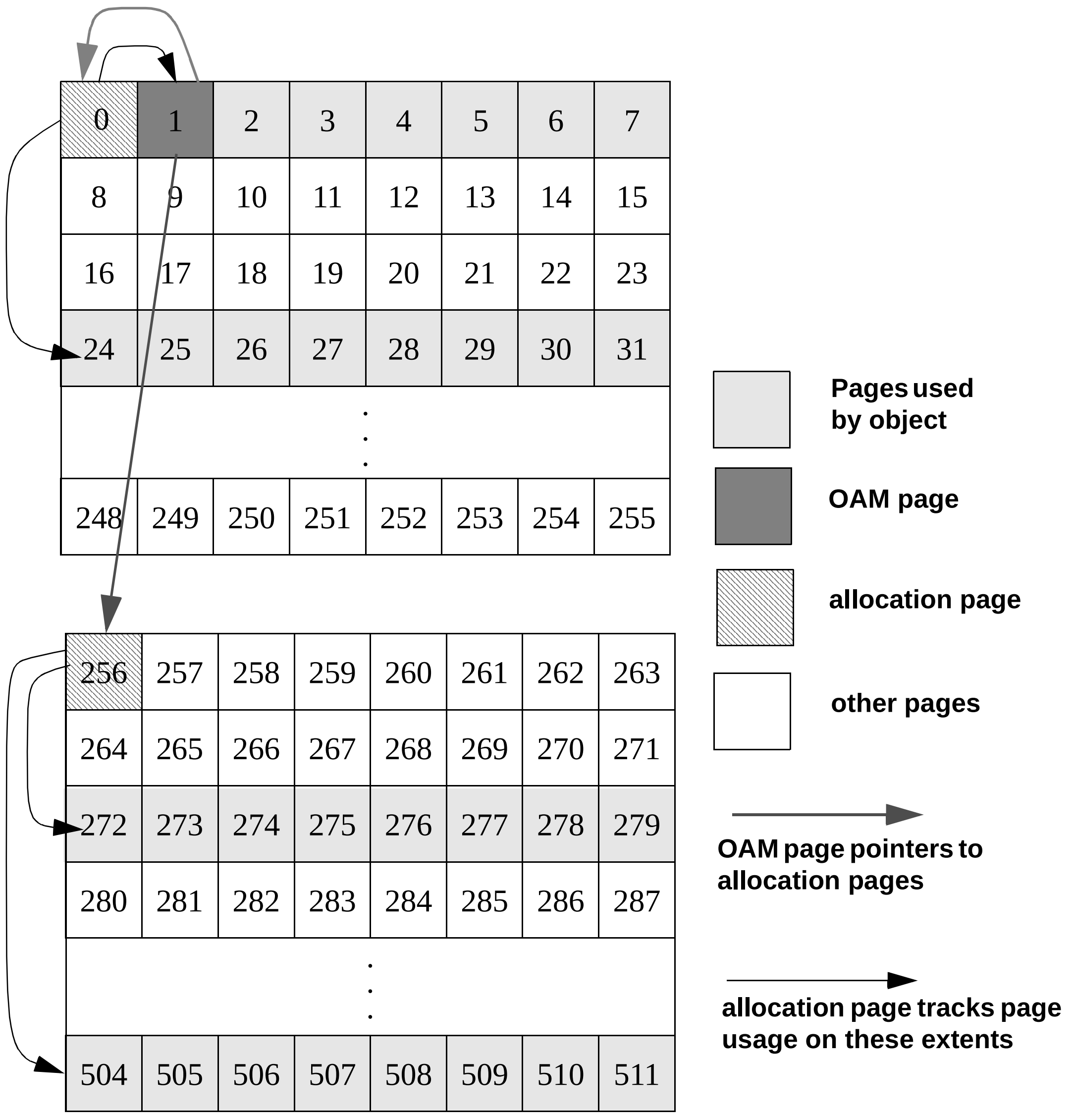
The OAM pages point to the allocation page for each allocation unit where the object uses space. The allocation pages, in turn, track the information about extent and page usage within the allocation unit. In other words, if the titles table is stored on extents 24 and 272, the OAM page for the titles table points to pages 0 and 256.

Figure 26-2 shows an object stored on 4 extents, numbered 0, 24, 272 and 504 for a server that uses 2K logical pages. The OAM is stored on the first page of the first segment. In this case, since the allocation page occupies page 0, the OAM is located on page 1.

This OAM points to two allocation pages: page 0 and page 256.

These allocation pages track the pages used in each extent used by all objects with storage space in the allocation unit. For the object in this example, it tracks the allocation and de-allocation of pages on extents 0, 24, 272, and 504.

Figure 26-2: OAM page and allocation page pointers

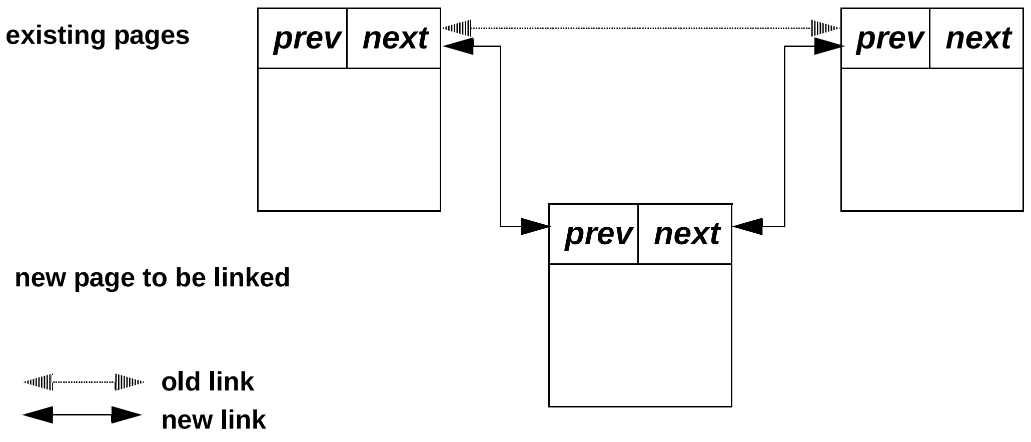


dbcc checkalloc and dbcc tablealloc examine this OAM page information, in addition to checking page linkage, as described in "Understanding page linkage" on page 765.

Understanding page linkage

After a page has been allocated to a table or an index, that page is linked with other pages used for the same object. Figure 26-3 illustrates this linking. Each page contains a header that includes the number of the page that precedes it (“prev”) and of the page that follows it (“next”). When a new page is allocated, the header information on the surrounding pages changes to point to that page. dbcc checktable and dbcc checkdb check page linkage. dbcc checkalloc, tablealloc, and indexalloc compare page linkage to information on the allocation page.

Figure 26-3: How a newly allocated page is linked with other pages



What checks can be performed with *dbcc*?

Table 26-1 summarizes the checks performed by the dbcc commands. Table 26-2 on page 778 compares the different dbcc commands.

Table 26-1: Comparison of checks performed by dbcc commands

Checks performed	checkstorage	checktable	checkdb	checkalloc	indexalloc	tablealloc	checkcatalog
Checks allocation of text valued columns	X						
Checks index consistency		X	X				
Checks index sort order		X	X				
Checks OAM page entries	X	X	X		X	X	
Checks page allocation	X			X	X	X	
Checks page consistency	X	X	X				
Checks pointer consistency	X	X	X				
Checks system tables							X
Checks text column chains	X	X	X	X			
Checks text valued columns	X	X	X				

Note You can run all dbcc commands except dbrepair and checkdb with the fix option while the database is active.

Only the table owner can execute dbcc with the checktable, fix_text, or reindex keywords. Only the Database Owner can use the checkstorage, checkdb, checkcatalog, checkalloc, indexalloc, and tablealloc keywords. Only a System Administrator can use the dbrepair keyword.

Checking consistency of databases and tables

The dbcc commands for checking the consistency of databases and tables are:

- dbcc checkstorage
- dbcc checktable
- dbcc checkdb

dbcc checkstorage

Use dbcc checkstorage to perform the following checks:

- Allocation of text valued columns
- Page allocation and consistency
- OAM page entries
- Pointer consistency
- Text valued columns and text column chains

The syntax for dbcc checkstorage is:

```
dbcc checkstorage [(dbname)]
```

where *dbname* is the name of the **target database** (the database to be checked).

dbcc checkstorage runs checks against the database on disk. If a corruption is only in memory, dbcc checkstorage may not detect the corruption. To ensure consistency between two dbcc checkstorage runs, run checkpoint before running dbcc checkstorage. However, doing so can turn a transient memory corruption into corruption on disk.

Advantages of using *dbcc checkstorage*

The dbcc checkstorage command:

- Combines many of the checks provided by the other dbcc commands
- Does not lock tables or pages for extended periods, which allows dbcc to locate errors accurately while allowing concurrent update activity
- Scales linearly with the aggregate I/O throughput
- Separates the functions of checking and reporting, which allows custom evaluation and report generation
- Provides a detailed description of space usage in the target database
- Records dbcc checkstorage activity and results in the dbccdb database, which allows trend analysis and provides a source of accurate diagnostic information

Comparison of *dbcc checkstorage* and other *dbcc* commands

dbcc checkstorage is different from the other dbcc commands in that it requires:

- The dbccdb database to store configuration information and the results of checks made on the target database. For more information, see "Preparing to use dbcc checkstorage" on page 789.
- At least two workspaces to use during the check operation. See "dbccdb workspaces" in Chapter 59, "dbccdb Tables" in the *Reference Manual*.
- System and stored procedures to help you prepare your system to use dbcc checkstorage and to generate reports on the data stored in dbccdb. See "Preparing to use dbcc checkstorage" on page 789, "Maintaining dbccdb" on page 800, and "Generating reports from dbccdb" on page 803.

dbcc checkstorage does not repair any faults. After you run dbcc checkstorage and generate a report to see the faults, you can run the appropriate dbcc command to repair the faults.

Understanding the *dbcc checkstorage* operation

The dbcc checkstorage operation consists of the following steps:

1. Inspection – dbcc checkstorage uses the device allocation and the segment definition of the database being checked to determine the level of parallel processing that can be used. dbcc checkstorage also uses the configuration parameters max worker processes and dbcc named cache to limit the level of parallel processing that can be used.
2. Planning – dbcc checkstorage generates a plan for executing the operation that takes advantage of the parallelism discovered in step 1.
3. Execution and optimization – dbcc checkstorage uses Adaptive Server worker processes to perform parallel checking and storage analysis of the target database. It attempts to equalize the work performed by each worker process and consolidates the work of under utilized worker processes. As the check operation proceeds, dbcc checkstorage extends and adjusts the plan generated in step 2 to take advantage of the additional information gathered during the check operation.

4 Reporting and control – during the check operation, dbcc checkstorage records in the dbccdb database all the faults it finds in the target database for later reporting and evaluation. It also records the results of its storage analysis in dbccdb. When dbcc checkstorage encounters a fault, it attempts to recover and continue the operation, but ends operations that cannot succeed after the fault. For example, a defective disk does not cause dbcc checkstorage to fail; however, check operations performed on the defective disk cannot succeed, so they are not performed.

If another session performs drop table concurrently, dbcc checkstorage might fail in the initialization phase. If this happens, run dbcc checkstorage again when the drop table process is finished.

Note See the *Troubleshooting and Error Message Guide* for information about dbcc checkstorage error messages.

Performance and scalability

dbcc checkstorage scales linearly with aggregate I/O throughput for a substantial performance improvement over dbcc checkalloc. The scaling property of dbcc checkstorage means that if the database doubles in size and the hardware doubles in capacity (realizable I/O throughput), the time required for a dbcc check remains unchanged. Doubling the capacity would typically mean doubling the number of disk spindles and providing sufficient additional I/O channel capacity, system bus capacity, and CPU capacity to realize the additional aggregate disk throughput.

Most of the checks performed by using dbcc checkalloc and dbcc checkdb, including text column chain verification, are achieved with a single check when you use dbcc checkstorage, thereby eliminating redundant check operations.

dbcc checkstorage checks the entire database, including unused pages, so execution time is relative to database size. Therefore, when you use dbcc checkstorage, there is not a large difference between checking a database that is nearly empty and checking one that is nearly full, as there is with the other dbcc commands.

Unlike the other dbcc commands, the performance of dbcc checkstorage does not depend heavily on data placement. Therefore, performance is consistent for each session, even if the data placement changes between sessions.

Because dbcc checkstorage does extra work to set up the parallel operation and records large amounts of data in dbccdb, the other dbcc commands are faster when the target database is small.

The location and allocation of the workspaces used by dbcc checkstorage can affect performance and scalability. For more information on how to set up the workspaces to maximize the performance and scalability of your system, see “dbcc workspaces” in Chapter 59, “dbccdb Tables” in the *Reference Manual*.

To run dbcc checkstorage and one of the system procedures for generating reports with a single command, use sp_dbcc_runcheck. For information on the report generating system procedures, see “Generating reports from dbccdb” on page 803.

dbcc checktable

dbcc checktable checks the specified table to see that:

- Index and data pages are linked correctly
- Indexes are sorted properly
- Pointers are consistent
- Data rows on each page have entries in the row-offset table; these entries match the locations for the data rows on the page
- Data rows on each page have entries in the row-offset table in the page that match their respective locations on the page
- Partition statistics for partitioned tables are correct

The syntax for dbcc checktable is:

```
dbcc checktable ({table_name | table_id}
    [, skip_ncindex] )
```

The skip_ncindex option allows you to skip checking the page linkage, pointers, and sort order on nonclustered indexes. The linkage and pointers of clustered indexes and data pages are essential to the integrity of your tables. You can drop and re-create nonclustered indexes if Adaptive Server reports problems with page linkage or pointers.

When checkstorage returns a fault code of 100035, and checkverify confirms that the spacebit fault is a hard fault, you can use dbcc checktable to fix the reported fault.

The following is the syntax, where *table_name* is the name of the table to repair:

```
dbcc checktable (table_name, fix_spacebits)
```

dbcc checktable can be used with the table name or the table's object ID. The sysobjects table stores this information in the name and id columns.

The following example shows a report on an undamaged table:

```
dbcc checktable(titles)
go
```

```
Checking titles
The total number of data pages in this table is 3.
Table has 18 data rows.
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator
(SA) role.
```

If the table is partitioned, dbcc checktable checks data page linkage and partition statistics for each partition. For example:

```
dbcc checktable(historytab)
go
```

```
Checking historytab
The total number of pages in partition 1 is 20.
The total number of pages in partition 2 is 17.
The total number of pages in partition 3 is 19.
The total number of pages in partition 4 is 17.
The total number of pages in partition 5 is 20.
The total number of pages in partition 6 is 16.
The total number of pages in partition 7 is 19.
The total number of pages in partition 8 is 17.
The total number of pages in partition 9 is 19.
The total number of pages in partition 10 is 16.

The total number of data pages in this table is 190.
Table has 1536 data rows.
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator
(SA) role.
```

To check a table that is not in the current database, supply the database name. To check a table owned by another object, supply the owner's name. You must enclose any qualified table name in quotes. For example:

```
dbcc checktable("pubs2.newuser.testtable")
```

dbcc checktable addresses the following problems:

- If the page linkage is incorrect, dbcc checktable displays an error message.
- If the sort order (sysindexes.soid) or character set (sysindexes.csid) for a table with columns with char or varchar datatypes is incorrect, and the table's sort order is compatible with Adaptive Server's default sort order, dbcc checktable corrects the values for the table. Only the binary sort order is compatible across character sets.

Note If you change sort orders, character-based user indexes are marked "read-only" and must be checked and rebuilt, if necessary. See Chapter 7, "Configuring Character Sets, Sort Orders, and Languages," for more information about changing sort orders.

- If data rows are not accounted for in the first OAM page for the object, dbcc checktable updates the number of rows on that page. This is not a serious problem. The built-in function rowcnt uses this value to provide fast row estimates in procedures like sp_spaceused.

You can improve dbcc checktable performance by using enhanced page fetching.

dbcc checkdb

dbcc checkdb runs the same checks as dbcc checktable on each table in the specified database. If you do not give a database name, dbcc checkdb checks the current database. dbcc checkdb gives similar messages to those returned by dbcc checktable and makes the same types of corrections.

The syntax for dbcc checkdb is:

```
dbcc checkdb [(database_name [, skip_ncindex]) ]
```

If you specify the optional skip_ncindex, dbcc checkdb does not check any of the nonclustered indexes on user tables in the database.

Checking page allocation

The dbcc commands that you use to check page allocation are:

- dbcc checkalloc

- dbcc indexalloc
- dbcc tablealloc

dbcc checkalloc

dbcc checkalloc ensures that:

- All pages are correctly allocated
- Partition statistics on the allocation pages are correct
- No page is allocated that is not used
- No page is used that is not allocated

The syntax for dbcc checkalloc is:

dbcc checkalloc [(*database_name* [, fix | nofix])]

If you do not provide a database name, dbcc checkalloc checks the current database.

With the fix option, dbcc checkalloc can fix all allocation errors that would otherwise be fixed by dbcc tablealloc and can also fix pages that remain allocated to objects that have been dropped from the database. Before you can use dbcc checkalloc with the fix option, you must put the database into single-user mode. For details on using the fix and no fix options, see “Correcting allocation errors using the fix | nofix option” on page 776.

dbcc checkalloc output consists of a block of data for each table, including the system tables and the indexes on each table. For each table or index, it reports the number of pages and extents used. Table information is reported as either INDID=0 or INDID=1. Tables without clustered indexes have INDID=0, as shown in the example report on the salesdetail table. Tables with clustered indexes have INDID=1. The report for these indexes includes information at both the data and index level, as shown in the example reports on titleauthor and titles. Nonclustered indexes are numbered consecutively, starting with INDID=2.

The following report on pubs2 shows the output for the salesdetail, titleauthor, and titles tables:

```
***************************************************************
TABLE: salesdetail              OBJID = 144003544
INDID=0  FIRST=297       ROOT=299        SORT=0
        Data level: 0.  3 Data  Pages in 1 extents.
INDID=2  FIRST=289       ROOT=290        SORT=1
```

```
        Indid     : 2.  3 Index Pages in 1 extents.
INDID=3  FIRST=465       ROOT=466        SORT=1
        Indid     : 3.  3 Index Pages in 1 extents.
TOTAL # of extents = 3
***************************************************************
TABLE: titleauthor              OBJID = 176003658
INDID=1  FIRST=433       ROOT=425        SORT=1
        Data level: 1.  1 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=305       ROOT=305        SORT=1
        Indid     : 2.  1 Index Pages in 1 extents.
INDID=3  FIRST=441       ROOT=441        SORT=1
        Indid     : 3.  1 Index Pages in 1 extents.
TOTAL # of extents = 4
***************************************************************
TABLE: titles           OBJID = 208003772
INDID=1  FIRST=417       ROOT=409        SORT=1
        Data level: 1.  3 Data  Pages in 1 extents.
        Indid     : 1.  1 Index Pages in 1 extents.
INDID=2  FIRST=313       ROOT=313        SORT=1
        Indid     : 2.  1 Index Pages in 1 extents.
TOTAL # of extents = 3
***************************************************************
```

dbcc indexalloc

dbcc indexalloc checks the specified index to see that:

- All pages are correctly allocated.
- No page is allocated that is not used.
- No page is used that is not allocated.

dbcc indexalloc is an index-level version of dbcc checkalloc, providing the same integrity checks on an individual index. You can specify either the table name or the table's object ID (the id column in sysobjects) and the index's indid in sysindexes. dbcc checkalloc and dbcc indexalloc include the index IDs in their output.

The syntax for dbcc indexalloc is:

dbcc indexalloc ({*table_name* | *table_id* }, *index_id*
    [, {full | optimized | fast | null}
    [, fix | nofix]])

If you want to use the fix or nofix option for dbcc indexalloc, you must also specify one of the report options (full, optimized, fast, or null). For details on using the fix and no fix options, see “Correcting allocation errors using the fix | nofix option” on page 776. For details on the reports, see “Generating reports with dbcc tablealloc and dbcc indexalloc” on page 776.

You can run sp_indsuspect to check the consistency of sort order in indexes and dbcc reindex to repair inconsistencies. For details see “Using sp_indsuspect to find corrupt indexes” on page 289 and “Rebuilding indexes after changing the sort order” on page 289.

dbcc tablealloc

dbcc tablealloc checks the specified user table to ensure that:

- All pages are correctly allocated.
- Partition statistics on the allocation pages are correct.
- No page is allocated that is not used.
- No page is used that is not allocated.

The syntax for dbcc tablealloc is:

```
dbcc tablealloc ({table_name | table_id}
    [, {full | optimized | fast | null}
    [, fix | nofix]])
```

You can specify either the table name or the table’s object ID from the id column in sysobjects.

If you want to use the fix or nofix options for dbcc tablealloc, you must also specify one of the report options (full, optimized, fast, or null). For details on using the fix and no fix options, see “Correcting allocation errors using the fix | nofix option” on page 776. For details on the reports, see “Generating reports with dbcc tablealloc and dbcc indexalloc” on page 776.

Correcting allocation errors using the *fix* | *nofix* option

You can use the fix | nofix option with dbcc checkalloc, dbcc tablealloc, and dbcc indexalloc. It specifies whether or not the command fixes the allocation errors in tables. The default for all user tables is fix. The default for all system tables is nofix.

Before you can use the fix option on system tables, you must put the database into single-user mode:

```
sp_dboption dbname, "single user", true
```

You can issue this command only when no one is using the database.

Output from dbcc tablealloc with fix displays allocation errors and any corrections that were made. The following example shows an error message that appears whether or not the fix option is used:

```
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt: The entry is missing from the OAM for
object id 144003544 indid 0 for allocation page 2560.
```

When you use fix, the following message indicates that the missing entry has been restored:

```
The missing OAM entry has been inserted.
```

The fix|nofix option works the same in dbcc indexalloc as it does in dbcc tablealloc.

Generating reports with *dbcc tablealloc* and *dbcc indexalloc*

You can generate three types of reports with dbcc tablealloc or dbcc indexalloc:

- full – produces a report containing all types of allocation errors. Using the full option with dbcc tablealloc gives the same results as using dbcc checkalloc at a table level.

- optimized – produces a report based on the allocation pages listed in the OAM pages for the table. When you use the optimized option, dbcc tablealloc does not report and cannot fix unreferenced extents on allocation pages that are not listed in the OAM pages. If you do not specify a report type, or if you specify null, optimized is the default.
- fast – produces an exception report of pages that are referenced but not allocated in the extent (2521-level errors); does not produce an allocation report.

For a comparison of speed, completeness, locking, and performance issues for these options and other dbcc commands, see Table 26-2 on page 778.

Checking consistency of system tables

dbcc checkcatalog checks for consistency within and between the system tables in a database. For example, it verifies that:

- Every type in syscolumns has a matching entry in systypes.
- Every table and view in sysobjects has at least one column in syscolumns.
- The last checkpoint in syslogs is valid.

It also lists the segments defined for use by the database.

The syntax for dbcc checkcatalog is:

```
dbcc checkcatalog [(database_name)]
```

If you do not specify a database name, dbcc checkcatalog checks the current database.

```
dbcc checkcatalog (testdb)
```

```
Checking testdb
The following segments have been defined for database 5 (database name testdb).
virtual start addr      size      segments
--------------------    ------    --------------------------
33554432                4096       0
                                   1
16777216                102        2
DBCC execution completed. If DBCC printed error messages, see your System
Administrator.
```

Strategies for using consistency checking commands

The following sections compare the performance of the dbcc commands, provide suggestions for scheduling and strategies to avoid serious performance impacts, and provide information about dbcc output.

Comparing the performance of *dbcc* commands

Table 26-2 compares the speed, thoroughness, the level of checking and locking, and performance implications of the dbcc commands. Remember that dbcc checkdb, dbcc checktable, and dbcc checkcatalog perform different types of integrity checks than dbcc checkalloc, dbcc tablealloc, and dbcc indexalloc. dbcc checkstorage performs a combination of the some of the checks performed by the other commands. Table 26-1 on page 766 shows which checks are performed by the commands.

Table 26-2: Comparison of the performance of dbcc commands

Command and option	Level	Locking and performance	Speed	Thorough -ness
checkstorage	Page chains and data rows for all indexes, allocation pages, OAM pages, device and partition statistics	No locking; performs extensive I/O and may saturate the system's I/O; can use dedicated cache with minimal impact on other caches	Fast	High
checktable checkdb	Page chains, sort order, data rows, and partition statistics for all indexes	Shared table lock; dbcc checkdb locks one table at a time and releases the lock after it finishes checking that table	Slow	High
checktable checkdb with skip_ncindex	Page chains, sort order, and data rows for tables and clustered indexes	Shared table lock; dbcc checkdb locks one table at a time and releases the lock after it finishes checking that table	Up to 40 percent faster than without skip_ncindex	Medium
checkalloc	Page chains and partition statistics	No locking; performs extensive I/O and may saturate the I/O calls; only allocation pages are cached	Slow	High
tablealloc full indexalloc full with full	Page chains	Shared table lock; performs extensive I/O; only allocation pages are cached	Slow	High
tablealloc indexalloc with optimized	Allocation pages	Shared table lock; performs extensive I/O; only allocation pages are cached	Moderate	Medium

Command and option	Level	Locking and performance	Speed	Thorough-ness
tablealloc indexalloc with fast	OAM pages	Shared table lock	Fast	Low
checkcatalog	Rows in system tables	Shared page locks on system catalogs; releases lock after each page is checked; very few pages cached	Moderate	Medium

Using large I/O and asynchronous prefetch

Some dbcc commands can use large I/O and asynchronous prefetch when these are configured for the caches used by the databases or objects to be checked.

dbcc checkdb and dbcc checktable use large I/O pools for the page chain checks on tables when the tables use a cache with large I/O configured. The largest I/O size available is used. When checking indexes, dbcc uses only 2K buffers.

The dbcc checkdb, dbcc checktable, and the dbcc allocation checking commands, checkalloc, tablealloc and indexalloc, use asynchronous prefetch when it is available for the pool in use. See "Setting limits for dbcc" on page 256 in *Performance and Tuning: Optimizer and Abstract Plans* for more information.

Cache binding commands and the commands to change the size and asynchronous prefetch percentages for pools are dynamic commands. If you use these dbcc commands during off-peak periods, when user applications experience little impact, you can change these settings to speed dbcc performance and restore the normal settings when dbcc checks are finished. See Chapter 19, "Configuring Data Caches," for information on these commands.

Scheduling database maintenance at your site

There are several factors that determine how often you should run dbcc commands and which ones you need to run.

Database use

If your Adaptive Server is used primarily between the hours of 8:00 a.m. and 5:00 p.m., Monday through Friday, you can run dbcc checks at night and on weekends so that the checks do not have a significant impact on your users. If your tables are not extremely large, you can run a complete set of dbcc commands fairly frequently.

dbcc checkstorage and dbcc checkcatalog provide the best coverage at the lowest cost, assure recovery from backups. You can run dbcc checkdb or dbcc checktable less frequently to check index sort order and consistency. This check does not need to be coordinated with any other database maintenance activity. Reserve object-level dbcc checks and those checks that use the fix option for further diagnosis and correction of faults found by dbcc checkstorage.

If your Adaptive Server is used 24 hours a day, 7 days a week, you may want to limit the resource usage of dbcc checkstorage by limiting the number of worker processes or by using application queues. If you decide not to use dbcc checkstorage, you may want to schedule a cycle of checks on individual tables and indexes using dbcc checktable, dbcc tablealloc, and dbcc indexalloc. At the end of the cycle, when all tables have been checked, you can run dbcc checkcatalog and back up the database. For information on using application queues, see Chapter 5, “Distributing Engine Resources,” in *Performance and Tuning: Basics*.

Some sites with 24-hour, high-performance demands run dbcc checks by:

- Dumping the database to tape
- Loading the database dump into a separate Adaptive Server to create a duplicate database
- Running dbcc commands on the duplicate database
- Running dbcc commands with the fix options on appropriate objects in the original database, if errors are detected that can be repaired with the fix options

The dump is a logical copy of the database pages; therefore, problems found in the original database are present in the duplicate database. This strategy is useful if you are using dumps to provide a duplicate database for reporting or some other purpose.

Schedule the use of dbcc commands that lock objects to avoid interference with business activities. For example, dbcc checkdb acquires locks on all objects in the database while it performs the check. You cannot control the order in which it checks the objects. If you are running an application that uses table4, table5, and table6, and running dbcc checkdb takes 20 minutes to complete, the application will be blocked from accessing these tables, even when the command is not checking them.

Backup schedule

The more often you back up your databases and dump your transaction logs, the more data you can recover in case of failure. You must decide how much data you are willing to lose in the event of a disaster and develop a dump schedule to support that decision.

After you schedule your dumps, decide how to incorporate the dbcc commands into that schedule. You do not have to perform dbcc checks before each dump; however, you may lose additional data if a corruption occurs in the dumps.

An ideal time to dump a database is after you run a complete check of that database using dbcc checkstorage and dbcc checkcatalog. If these commands find no errors in the database, you know that your backup contains a clean database. You can correct problems that occur after loading the dump by reindexing. Use dbcc tablealloc or indexalloc on individual tables and indexes to correct allocation errors reported by dbcc checkalloc.

Size of tables and importance of data

Answer the following questions about your data:

- How many tables contain highly critical data?
- How often does that data change?
- How large are those tables?

dbcc checkstorage is a database-level operation. If only a few tables contain critical data or data that changes often, you may want to run the table- and index-level dbcc commands more frequently on those tables than you run dbcc checkstorage on the entire database.

Understanding the output from *dbcc* commands

dbcc checkstorage stores the results in the dbccdb database. You can print a variety of reports from this database. For details, see “dbcc checkstorage” on page 767.

The output of most other dbcc commands includes information that identifies the objects being checked and error messages that indicate any problems, the command finds in the object. When you run dbcc tablealloc and dbcc indexalloc with fix, the output also indicates the repairs that the command makes.

The following example shows dbcc tablealloc output for a table with an allocation error:

```
dbcc tablealloc(table5)
```

```
Information from sysindexes about the object being checked:
TABLE: table5            OBJID = 144003544
INDID=0  FIRST=337       ROOT=2587       SORT=0

Error message:
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt: The entry is missing from the OAM for object id
144003544 indid 0 for allocation page 2560.

Message indicating that the error has been corrected:

The missing OAM entry has been inserted.
        Data level: 0.  67 Data  Pages in 9 extents.

dbcc report on page allocation:

TOTAL # of extents = 9
Alloc page 256 (# of extent=1 used pages=8 ref pages=8)
EXTID:560 (Alloc page: 512) is initialized.  Extent follows:
NEXT=0 PREV=0 OBJID=144003544 ALLOC=0xff DEALL=0x0 INDID=0 STATUS=0x0
Alloc page 512 (# of extent=2 used pages=8 ref pages=8)
Page 864 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x1)
Page 865 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x3)
Page 866 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x7)
Page 867 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0xf)
Page 868 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x1f)
Page 869 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x3f)
Page 870 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0x7f)
Page 871 allocated (Alloc page: 768 Extent ID: 864 Alloc mask: 0xff)
Alloc page 768 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1024 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1280 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1536 (# of extent=1 used pages=8 ref pages=8)
Alloc page 1792 (# of extent=1 used pages=8 ref pages=8)
Alloc page 2048 (# of extent=1 used pages=8 ref pages=8)
(Other output deleted.)

Information on resources used:
Statistical information for this run follows:
Total # of pages read = 68
Total # of pages found cache = 68
Total # of physical reads = 0
Total # of saved I/O = 0

Message printed on completion of dbcc command:
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

Errors generated by database consistency problems

Errors generated by database consistency problems encountered by dbcc checkstorage are documented in the dbcc_types table. Most are in the ranges 5010–5024 and 100,000–100,038. For information on specific errors, see “dbcc_types” on page 1388 of the *Reference Manual*. dbcc checkstorage records two kinds of faults: soft and hard. For information, see “Comparison of soft and hard faults” on page 784.

Errors generated by database consistency problems encountered by dbcc commands other than dbcc checkstorage usually have error numbers from 2500 to 2599 or from 7900 to 7999. These messages, and others that can result from database consistency problems (such as Error 605), may include phrases like “Table Corrupt” or “Extent not within segment.”

Some messages indicate severe database consistency problems; others are not so urgent. A few may require help from Sybase Technical Support, but most can be solved by:

- Running dbcc commands that use the fix option
- Following the instructions in the *Error Messages and Troubleshooting Guide*, which contains step-by-step instructions for resolving many database errors found by dbcc

Whatever techniques are required to solve the problems, the solutions are much easier when you find the problem soon after the occurrence of the corruption or inconsistency. Consistency problems can exist on data pages that are not used frequently, such as a table that is updated only monthly. dbcc can find, and often fix, these problems for you.

Comparison of soft and hard faults

When dbcc checkstorage finds a fault in the target database, it is recorded in the dbcc_faults table as either a **soft fault** or a **hard fault**. The following sections describe the two kinds of faults. For more information, see "Verifying faults with dbcc checkverify" on page 785.

Soft faults

A *soft fault* is an inconsistency in Adaptive Server that is usually not persistent. Most soft faults result from temporary inconsistencies in the target database caused by users updates to the database during dbcc checkstorage or when dbcc checkstorage encounters data definition language (DDL) commands. These faults are not repeated when you run the command a second time. You can reclassify soft faults by comparing the results of the two executions of dbcc checkstorage or by running dbcc tablealloc and dbcc checktable after dbcc checkstorage finds soft faults.

If the same soft faults occur in successive executions of dbcc checkstorage, they are "persistent" soft faults, and may indicate a corruption. If you execute dbcc checkstorage in single-user mode, the soft faults reported are "persistent" soft faults. You can resolve these faults by using sp_dbcc_differentialreport or by running dbcc tablealloc and dbcc checktable. If you use the latter two commands, you need to check only the tables or indexes that exhibited the soft faults.

Hard faults

A *hard fault* is a persistent corruption of Adaptive Server that cannot be corrected by restarting Adaptive Server. Not all hard faults are equally severe. For example, each of the following situations cause a hard fault, but the results are different:

- A page that is allocated to a nonexistent table minimally reduces the available disk storage.

- A table with some rows that are unreachable by a scan might return the wrong results.
- A table that is linked to another table causes the query to stop.

Some hard faults can be corrected by simple actions such as truncating the affected table. Others can be corrected only by restoring the database from a backup.

Verifying faults with *dbcc checkverify*

dbcc checkverify examines the results of the most recent checkstorage operation and reclassifies each soft fault as either a hard fault or an insignificant fault. checkverify acts as a second filter to remove spurious faults from the checkstorage results.

How *dbcc checkverify* works

checkverify reads the recorded faults from dbcc_faults and resolves each soft fault through a procedure similar to that used by the checkstorage operation.

Note checkverify locks the table against concurrent updates, which ensures that the soft faults are reclassified correctly. checkverify does not find errors that have occurred since the last run of checkstorage.

checkverify records information in the dbcc_operation_log and dbcc_operation_results tables the same way that checkstorage does. The recorded value of opid is the same as the opid of the last checkstorage operation. checkverify updates the status column in the dbcc_faults table and inserts a row in the dbcc_fault_params table for the faults it processes.

checkverify does not use the scan or text workspaces.

Each fault found by checkstorage is verified by checkverify as one of the following:

- A hard fault classified as such by checkstorage.

- A soft fault reclassified as hard by checkverify because concurrent activity was ruled out as the cause.
- A soft fault confirmed to be soft by checkverify. Some soft faults that appear when there is no concurrent activity in the database do not represent a significant hazard and are not reclassified as hard. A soft fault is not reclassified if it is informational only and not a corruption.
- A soft fault reclassified as insignificant because it can be attributed to concurrent activity or because subsequent activity masked the original inconsistency.

A fault that is assigned code 100011 (text pointer fault) by checkstorage is verified as hard if the text column has a hard fault. If it does not, it is reclassified as soft.

A fault that is assigned code 100016 (page allocated but not linked) by checkstorage is verified as hard if the same fault appears in two successive checkstorage operations. Otherwise, it is reclassified as soft.

When a fault that is assigned code 100035 (spacebits mismatch) by checkstorage is verified as hard, you can repair it by using dbcc checktable.

When checkverify confirms hard faults in your database, follow the same procedures as you did in previous versions of Adaptive Server to correct the faults.

checkverify classifies the following fault codes as soft faults:

- 100020 – check aborted
- 100025 – row count fault
- 100028 – page allocation off current segment

When to use *dbcc checkverify*

You can verify persistent faults by running checkverify anytime after running checkstorage, even after an extended period of hours or days. However, when deciding your schedule, keep in mind that the database state changes over time, and the changes can mask both soft faults and hard faults.

For example, a page that is linked to a table but not allocated is a hard fault. If the table is dropped, the fault is resolved and masked. If the page is allocated to another table, the fault persists but its signature changes. The page now appears to be linked to two different tables. If the page is reallocated to the same table, the fault appears as a corrupt page chain.

Persistent faults that are corrected by a subsequent database change usually do not pose an operational problem. However, detecting and quickly verifying these faults may locate a source of corruption before more serious problems are encountered or before the signature of the original fault changes. For this reason, Sybase recommends that you run checkverify as soon as possible after running dbcc checkstorage.

Note When checkstorage is executed with the target database in single-user mode, there will be no soft faults and no need to execute checkverify.

---

checkverify runs only one time for each execution of checkstorage. However, if checkverify is interrupted and does not complete, you can run it again. The operation resumes from where it was interrupted.

checkverify may take a long time to complete when processing very large databases. During this time, checkverify does not provide you with any indication of when it will finish. To get a status for the checkverify, use the command_status_reporting command. When the check_status_reporting is on, the checkverify will give progress status reports. The default value is off.

To turn the option 'on', the following command is issued:

```
set command_status_reporting on
```

The results will appear as:

```
1> dbcc checkverify (pubs2)
2> go
Verifying faults for’pubs2’.
Verifying faults for table 't1'. The total number of tables to verify is 5. This
is table number 1.
Verifying faults for table 't2'. The total number of tables to verify is 5. This
is table number 2.
Verifying faults for table 't3'. The total number of tables to verify is 5. This
is table number 3.
Verifying faults for table 't4'. The total number of tables to verify is 5. This
is table number 4.
Verifying faults for table 't5'. The total number of tables to verify is 5. This
is table number 5.
```

```
DBCC CHECKVERIFY for database 'pubs2' sequence 4 completed at Apr  9 2003
2:40PM. 72 suspect conditions were resolved as faults, and 11 suspect conditions
were resolved as harmless. 0 objects could not be checked.
```

How to use *dbcc checkverify*

The syntax is:

dbcc checkverify(*dbname*)

where *dbname* is the database for which you want to verify checkstorage results.

checkverify operates on the results of the last completed checkstorage operation for the specified database only.

When the checkverify operation is complete, Adaptive Server returns the following message:

```
DBCC checkverify for database name, sequence
n completed at date time. n suspect conditions
resolved as faults and n resolved as innocuous.
n checks were aborted.
```

You can run checkverify automatically after running checkstorage by using sp_dbcc_runcheck.

Dropping a damaged database

Use dbcc dbrepair dropdb from the master database to drop a damaged database. No users, including the user running dbrepair, can be using the database when it is dropped.

The syntax for dbcc dbrepair is:

dbcc dbrepair (*database_name*, dropdb)

The Transact-SQL command drop database does not work on a database that cannot be recovered or used.

Preparing to use *dbcc checkstorage*

Before you can use dbcc checkstorage, you must configure Adaptive Server and set up the dbccdb database. Table 26-3 summarizes the steps and commands in the order you should use them. Each action is described in detail in the following sections.

Note The examples in this section assume a server that uses 2K logical pages.

Warning! Do not attempt to perform the actions or use the commands in Table 26-3 before you read the information in the referenced section. You must understand the consequences of each action before you make any changes.

Table 26-3: Tasks for preparing to use dbcc checkstorage

For this action	See	Use this command
1. Obtain recommendations for database size, devices (if dbccdb does not exist), workspace sizes, cache size, and the number of worker processes for the target database.	“Planning resources” on page 790 “Planning workspace size” on page 792	use master sp_plan_dbccdb
2. If necessary, adjust the number of worker processes that Adaptive Server uses.	“Configuring worker processes” on page 794	sp_configure number of worker processes memory per worker process
3. *Optional* – create a named cache for dbcc.	“Setting up a named cache for dbcc” on page 795	sp_cacheconfig
4. Configure a your buffer pool.	“Configuring an 8-page I/O buffer pool” on page 796	sp_poolconfig
5. If dbccdb already exists, drop it and all associated devices before creating a new dbccdb database.		drop database
6. Initialize disk devices for the dbccdb data and the log.	“Allocating disk space for dbccdb” on page 797	disk init
7. *Optional* – create dbccdb on the data disk device.		create database
8. *Optional* – add disk segments.	“Segments for workspaces” on page 797	use dbccdb

For this action	See	Use this command
9. Populate the dbccdb database and install dbcc stored procedures.		`isql -Usa -P -i $SYBASE/scripts/installdbccdb`
10. Create the workspaces.	"dbccdb workspaces" in Chapter 59, "dbccdb Tables" in the *Reference Manual*	sp_dbcc_createws

11. Update configuration values.	"Updating the dbcc_config table" on page 800	`sp_dbcc_updateconfig`
		`max worker processes`
		`dbcc named cache`
		`scan workspace`
		`text workspace`
		`OAM count threshold`
		`IO error abort`
		`linkage error abort`

Note dbcc checkstorage runs its checks against the database on disk. If the corruption is only in memory, dbcc may not detect it. To ensure consistency between two sequential dbcc checkstorage commands, first run a checkpoint. However, this can have the side-effect of turning a transient memory corruption into corruption on disk

Planning resources

Selecting the appropriate device and size for dbccdb is critical to the performance of dbcc checkstorage operations. sp_plan_dbccdb provides configuration recommendations or facts for the specified target database depending on whether dbccdb exists or not. You use this information to configure Adaptive Server and set up the dbccdb database.

Examples of *sp_plan_dbccdb* output

If dbccdb does not exist, sp_plan_dbccdb returns:

- Minimum size for dbccdb
- Devices that are suitable for dbccdb
- Minimum sizes for the scan and text workspaces
- Minimum cache size

- Number of worker processes

The values recommended for the cache size are approximate because the optimum cache size for dbccdb depends on the pattern of the page allocation in the target database. The following example from a server that uses 2K logical pages shows the output of sp_plan_dbccdb for the pubs2 database when dbccdb does not exist:

```
sp_plan_dbccdb pubs2
```

```
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process count are:

dbname         scan ws    text ws    cache    process count

pubs2          64K         64K        640K     1
```

If dbccdb already exists, sp_plan_dbccdb returns:

- Minimum size for dbccdb
- Size of existing dbccdb database
- Minimum sizes for the scan and text workspaces
- Minimum cache size
- Number of worker processes

The following example shows the output of sp_plan_dbccdb for the pubs2 database when dbccdb already exists:

```
sp_plan_dbccdb pubs2
```

```
Recommended size for dbccdb database is 23MB (data = 21MB, log = 2MB).

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process count are:

dbname                            scan ws     text ws     cache    process count
```

```
pubs2                            64K        48K        640K     1
```

If you plan to check more than one database, use the name of the largest one for the target database. If you do not provide a target database name, sp_plan_dbccdb returns configuration values for all databases listed in master..sysdatabases, as shown in the following example:

```
sp_plan_dbccdb
```

```
Recommended size for dbccdb is 4MB.

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process count are:

dbname           scan ws    text ws    cache    process count

master            64K        64K        640K     1
tempdb            64K        64K        640K     1
model             64K        64K        640K     1
sybsystemprocs  384K       112K       1280K     2
pubs2             64K        64K        640K     1
pubs3             64K        64K        640K     1
pubtune          160K        96K       1280K     2
sybsecurity       96K        96K       1280K     2
dbccdb           112K        96K       1280K     2
```

For more information, see “sp_plan_dbccdb” in the *Reference Manual*.

Planning workspace size

Two workspaces are required for dbccdb: scan and text. Space requirements for the workspaces depend on the size of the largest database that will be checked. To run concurrent dbcc checkstorage operations, you need to set up additional workspaces.

Determining the sIze for the largest database to be checked

Different databases can use the same workspaces. Therefore, the workspaces must be large enough to accommodate the largest database with which they will be used.

Note sp_plan_dbccdb suggests workspace sizes – the following details for determining the workspace size are provided for background information only.

Each page in the target database is represented by one 18-byte row in the scan workspace. This workspace should be approximately 1.1 percent of the target database size.

Every non-null text column in the target database inserts a 22-byte row in the text workspace. If there are *n* non-null text columns in the target database, the size of the text workspace must be at least (22 * *n*) bytes. The number of non-null text columns is dynamic, so allocate enough space for the text workspace to accommodate future demands. The minimum space required for the text workspace is 24 pages.

Number of workspaces that can be used concurrently

You can configure dbccdb to run dbcc checkstorage concurrently on multiple databases. This is possible only when the second and subsequent dbcc checkstorage operations have their own dedicated resources. To perform concurrent dbcc checkstorage operations, each operation must have its own scan and text workspaces, worker processes, and reserved cache.

The total space requirement for workspaces depends on whether the user databases are checked serially or concurrently. If dbcc checkstorage operations are run serially, the largest scan and text workspaces can be used for all user databases. If dbcc checkstorage operations are run concurrently, then dbccdb should be set to accommodate the largest workspaces that will be used concurrently. You can determine the workspace sizes by adding the sizes of the largest databases that will be checked concurrently.

For more information, see “dbccdb workspaces” in Chapter 59, “dbccdb Tables” in the *Reference Manual*.

Configuring Adaptive Server for *dbcc checkstorage*

This section provides information on configuring Adaptive Server for dbcc checkstorage.

Configuring worker processes

The following parameters affect dbcc checkstorage:

- max worker processes – set this parameter with sp_dbcc_updateconfig. It updates the value of max worker processes in the dbcc_config table for each target database.
- number of worker processes – set this configuration parameter with sp_configure. It updates the *server_name.cfg* file.
- memory per worker process – set this configuration parameter with sp_configure. It updates the *server_name.cfg* file.

After changing the value of the sp_configure parameters, you must restart Adaptive Server for the change to take effect. For details, see Chapter 4, "Setting Configuration Parameters."

max worker processes specifies the maximum number of worker processes used by dbcc checkstorage for each target database, while number of worker processes specifies the total number of worker processes supported by Adaptive Server. Worker processes are not dedicated to running dbcc checkstorage operations.

Set the value for number of worker processes high enough to allow for the number of processes specified by max worker processes. A low number of worker processes reduces the performance and resource consumption of dbcc checkstorage. dbcc checkstorage will not use more processes than the number of database devices used by the database. Cache size, CPU performance, and device sizes might suggest a lower worker processes count. If there are not enough worker processes configured for Adaptive Server, dbcc checkstorage will not run.

maximum parallel degree and maximum scan parallel degree have no effect on the parallel functions of dbcc checkstorage. When maximum parallel degree is set to 1, parallelism in dbcc checkstorage is not disabled.

dbcc checkstorage requires multiple processes, so number of worker processes must be set to at least 1 to allow for a parent process and a worker process.

sp_plan_dbccdb recommends values for the number of worker processes, depending on database size, number of devices, and other factors. You can use smaller values to limit the load on your system. dbcc checkstorage may use fewer worker processes than sp_plan_dbccdb recommends or fewer than you configure.

Using more worker processes does not guarantee faster performance. The following scenario describes the effects of two different configurations:

An 8GB database has 4GB of data on disk A and 0.5GB of data on each of the disks B, C, D, E, F, G, H, and I.

With 9 worker processes active, the time it takes to run dbcc checkstorage is 2 hours, which is the time it takes to check disk A. Each of the other 8 worker processes finishes in 15 minutes and waits for the disk A worker process to finish.

With 2 worker processes active, the time it takes to run dbcc checkstorage is still 2 hours. The first worker process processes disk A and the other worker process processes disks B, C, D, E, F, G, H, and I. In this case, there is no waiting, and resources are used more efficiently.

memory per worker process specifies the total memory allocation for worker processes support in Adaptive Server. The default value is adequate for dbcc checkstorage.

Setting up a named cache for *dbcc*

If you use a named cache for dbcc checkstorage, you might need to adjust the Adaptive Server configuration parameters.

During a dbcc checkstorage operation, the workspaces are temporarily bound to a cache which is also used to read the target database. Using a named cache that is dedicated to dbcc minimizes the impact of the database check on other users and improves performance. You can create a separate cache for each dbcc checkstorage operation that will be run concurrently, or you can create one cache that is large enough to fit the total requirements of the concurrent operations. The size required for optimum performance depends on the size of the target database and distributions of data in that database. dbcc checkstorage requires a minimum of 640K of buffers (each buffer is one extent) per worker process in the named cache.

For best performance, assign most of the dedicated cache to the buffer pool and do not partition the cache. The recommended cache size is the minimum size for the buffer pool. Add the size of the one page pool to this value.

If you dedicate a cache for dbcc checkstorage, the command does not require more than the minimum one page buffer pool. If the cache is shared, you can improve the performance of dbcc checkstorage by increasing the buffer pool size before running the operation, and reducing the size after the operation is complete. The buffer pool requirements are the same for a shared cache. However, while a shared cache may meet the size requirement, other demands on the cache might limit the buffer availability to dbcc checkstorage and greatly impact the performance of both checkstorage and Adaptive Server as a whole.

Warning! Do not use cache partitions in a cache being used for dbcc checkstorage.

To configure Adaptive Server with a named cache for dbcc checkstorage operations, use sp_cacheconfig and sp_poolconfig. See Chapter 19, “Configuring Data Caches.”

Configuring an 8-page I/O buffer pool

dbcc checkstorage requires a I/O buffer pool of one extent. Use sp_poolconfig to configure the pool size and verify that the pool has been configured properly. The pool size is stored in the dbcc_config table.

The following example shows how to use sp_poolconfig to set a 16K buffer pool for “master_cache” on a server configured for 2K logical pages. The named cache that is created for the master database.

```
1> sp_poolconfig "master_cache", "1024K", "16K"
2> go

(return status = 0)
```

The following example shows that the buffer pool for the private cache “master_cache” is set:

```
    1> sp_poolconfig "master_cache"
    2> go
Cache Name     Status    Type     Config Value Run Value
------------- --------- -------- ------------ ------------
master_cache   Active    Mixed         2.00 Mb      2.00 Mb
```

```
                                ------------ ------------
                         Total        2.00 Mb      2.00 Mb
=================================================================
Cache: master_cache,   Status: Active,   Type: Mixed
   Config Size: 2.00 Mb,   Run Size: 2.00 Mb
   Config Replacement: strict LRU,   Run Replacement: strict LRU
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
2 Kb     512 Kb    0.00 Mb      1.00 Mb      10
16 Kb    192 Kb    1.00 Mb      1.00 Mb      10
(return status = 0)
```

For more information on sp_poolconfig, see the *Reference Manual*.

Allocating disk space for *dbccdb*

Additional disk storage is required for the dbccdb database. Because dbcc checkstorage uses dbccdb extensively, you should place dbccdb on a device that is separate from other database devices.

Note Do not create dbccdb on the master device. Make sure that the log devices and data devices for dbccdb are separate.

Segments for workspaces

By dedicating segments for workspaces, you can control the workspace placement and improve the performance of dbcc checkstorage performance. When you dedicate new segments for the exclusive use of workspaces, be sure to unmap the devices attached to these segments from the default segment with sp_dropsegment.

Creating the *dbccdb* database

v **Creating the *dbccdb* database**

1. Run sp_plan_dbccdb in the master database to obtain recommendations for database size, devices, workspace sizes, cache size, and the number of worker processes for the target database. For example, suppose you run sp_plan_dbccdb with pubs2 as the target database when dbccdb did not exist:

   ```
   use master
   ```

```
go
sp_plan_dbccdb pubs2
go
```

The following output appears:

```
Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process count are:

dbname         scan ws    text ws    cache    process count

pubs2          64K         64K        640K     1
```

For details on the information provided by sp_plan_dbccdb, see "Planning resources" on page 790.

2 If dbccdb already exists, drop it and all associated devices before creating a new dbccdb database:

```
use master
go
if exists (select * from master.dbo.sysdatabases
     where name = "dbccdb")
begin
     print "+++ Dropping the dbccdb database"
     drop database dbccdb
end
go
```

3 Use disk init to initialize disk devices for the dbccdb data and the log:

```
use master
go
disk init
   name = "dbccdb_dat",
   physname = "/remote/disks/masters/",
   size = "4096K"
go
disk init
```

```
   name = "dbccdb_log",
   physname = "/remote/disks/masters/",
size = "1024K"
go
```

4 Use create database to create dbccdb on the data disk device that you initialized in step 3:

```
use master
go
```

```
create database dbccdb
   on dbccdb_dat = 6
   log on dbccdb_log = 2
go
```

5 *Optional* – add segments for the scan and text workspaces to the dbccdb data device:

```
use dbccdb
go
sp_addsegment scanseg, dbccdb, dbccdb_dat
go
sp_addsegment textseg, dbccdb, dbccdb_dat
go
```

6 Create the tables for dbccdb and initialize the dbcc_types table:

```
isql -Ujms -P***** -iinstalldbccdb
```

The installdbccdb script checks for the existence of the database before it attempts to create the tables. It creates only those tables that do not already exist in dbccdb. If any of the dbccdb tables become corrupted, remove them with drop table, and then use installdbccdb to re-create them.

7 Create and initialize the scan and text workspaces:

```
use dbccdb
go
sp_dbcc_createws dbccdb, scanseg, scan_pubs2,
scan, "10M"
go
sp_dbcc_createws dbccdb, textseg, text_pubs2,
text, "10M"
go
```

When you have finished installing dbccdb, you must update the dbcc_config table.

Updating the *dbcc_config* table

Use sp_dbcc_updateconfig to initialize the dbcc_config table for the *target database*. You must update each dbcc parameter separately for each target database, as shown in the following example:

```
use dbccdb
go

sp_dbcc_updateconfig pubs2,"max worker processes", "4"
go

sp_dbcc_updateconfig pubs2, "dbcc named cache", pubs2_cache, "10K"
go

sp_dbcc_updateconfig pubs2, "scan workspace", scan_pubs2
go

sp_dbcc_updateconfig pubs2, "text workspace", text_pubs2
go

sp_dbcc_updateconfig pubs2, "OAM count threshold", "5"
go

sp_dbcc_updateconfig pubs2, "IO error abort", "3"
go

sp_dbcc_updateconfig pubs2,"linkage error abort", "8"
go
```

You can now use dbcc checkstorage to check your databases. For descriptions of the dbcc parameters, see type code values 1 through 9 in "dbcc_types" in Chapter 59, "dbccdb Tables" in the *Reference Manual*.

Maintaining *dbccdb*

You will occasionally need to perform maintenance tasks on dbccdb.

- Reevaluate and update the configuration using:
  - sp_dbcc_evaluatedb – recommends values for configuration parameters using the results of previous dbcc checkstorage operations.

    - sp_dbcc_updateconfig – updates the configuration parameters for the specified database.
- Clean up obsolete data in dbccdb:
    - sp_dbcc_deletedb – deletes all the information on the specified database from dbccdb.
    - sp_dbcc_deletehistory – deletes the results of the dbcc checkstorage operations on the specified database from dbccdb.
- Remove unnecessary workspaces.
- Perform consistency checks on dbccdb itself.

The following sections describe the maintenance tasks in greater detail.

Reevaluating and updating *dbccdb* configuration

If the characteristics of user databases change, use sp_dbcc_evaluatedb to reevaluate the current dbccdb configuration and recommend more suitable values.

The following changes to user databases might affect the dbccdb configuration, as follows:

- When a user database is created, deleted or altered, the size of the workspaces and named cache, or the number of worker threads stored in the dbcc_config table might be affected.
- Changes in the named cache size or worker process count for dbcc_checkstorage may require you to reconfigure buffer cache and worker processes.

If the results of dbcc checkstorage operations are available for the target database, use sp_dbcc_evaluatedb to determine new configuration values. sp_dbcc_configreport also reports the configuration parameters for the specified database.

Use sp_dbcc_updateconfig to add new databases to the dbcc_config table and to change the configuration values in dbcc_config to reflect the values recommended by sp_dbcc_evaluatedb.

Cleaning up *dbccdb*

Adaptive Server stores data generated by dbcc checkstorage in dbccdb. You should periodically clean up dbccdb by using sp_dbcc_deletehistory to delete data for the target database that was created before the date you specify.

When you delete a database, you should also delete from dbccdb all configuration information and dbcc checkstorage results related to that database. Use sp_dbcc_deletedb to delete all database information from dbccdb.

Removing workspaces

You may need to remove unnecessary workspaces. In dbccdb, issue:

```
drop table workspace_name
```

Performing consistency checks on *dbccdb*

The limited update activity in the dbccdb tables should make corruption less frequent. Two signs of corruption in dbccdb are:

- Failure of dbcc checkstorage during the initialization phase, as it evaluates the work that needs to be performed, or during the completion phase, when it records its results
- Loss of information about faults resulting from corruption in the recorded faults, found by dbcc checkstorage

A severe corruption in dbccdb may cause dbcc checkstorage to fail. For dbcc checkstorage to locate severe corruptions in dbccdb, you can create an alternate database, dbccalt, which you use only for checking dbccdb. Create dbccalt using the same process that you used to create dbccdb as described in “Preparing to use dbcc checkstorage” on page 789.

If no free devices are available for dbccalt, you can use any device that is not used by the master database or dbccdb.

dbcc checkstorage and the dbcc system procedures function the same with dbccalt as they do with dbccdb. When the target database is dbccdb, dbcc checkstorage uses dbccalt, if it exists. If dbccalt does not exist, dbccdb can be checked using itself as the management database. If the target database is dbccdb and dbccalt exists, the results of dbcc checkstorage operations on dbccdb are stored in dbccalt. If dbccalt does not exist, the results are stored in dbccdb itself.

Alternatively, dbcc checkalloc and dbcc checktable can be used to check dbccdb.

If dbccdb becomes corrupted, you can drop it and re-create it or load an older version from a backup. If you drop it, some of its diagnostic history will be lost.

Generating reports from *dbccdb*

Several dbcc stored procedures are provided with dbccdb so that you can generate reports from the data in dbccdb.

To report a summary of *dbcc checkstorage* operations

sp_dbcc_summaryreport reports all dbcc checkstorage operations that were completed for the specified database on or before the specified date. The following example shows output from this command:

```
sp_dbcc_summaryreport
```

```
DBCC Operation : checkstorage
Database Name      Start time          End Time   Operation ID
         Hard Faults Soft Faults Text Columns Abort Count
         User Name
------------------ ------------------- ---------- ------------
         ----------- ----------- ----------- -----------
         ------------------------------
sybsystemprocs     05/12/1997 10:54:45  10:54:53             1
         0           0            0           0
     sa
sybsystemprocs     05/12/1997 11:14:10  11:14:19             2
         0           0            0           0
     sa
```

For details, see Chapter 10, “dbcc Stored Procedures,” in the *Reference Manual*.

To report configuration, statistics and fault information

sp_dbcc_fullreport runs these reports in the order shown:

- sp_dbcc_summaryreport – for an example, see “To report a summary of dbcc checkstorage operations” on page 803.
- sp_dbcc_configreport – for an example, see “To see configuration information for a target database” on page 804.
- sp_dbcc_statisticsreport – for an example, see “To report statistics information from dbcc_counter” on page 806.
- sp_dbcc_faultreport short – for an example, see “To report faults found in a database object” on page 805.

To see configuration information for a target database

Use sp_dbcc_configreport to generate a report of the configuration information for a target database. The following example shows output from this command:

```
sp_dbcc_configreport
```

```
Reporting configuration information of database sybsystemprocs.

Parameter Name             Value                          Size

database name              sybsystemprocs                 51200K
dbcc named cache           default data cache             1024K
text workspace             textws_001 (id = 544004969)    128K
scan workspace             scanws_001 (id = 512004855)    1024K
max worker processes       1
operation sequence number  2
```

To compare results of *dbcc checkstorage* operations

sp_dbcc_differentialreport compares the results of the dbcc checkstorage operations completed for the specified database object on the specified dates. The following example shows output from this command:

```
sp_dbcc_differentialreport master, sysprocedures,
checkstorage, "01/01/96", "01/02/96"
```

```
The following changes in dbcc counter values for the object "sysprocedures" in
database master have been noticed between 01/01/96 and 01/02/96.

Description        Date1   Date2

pages used           999    1020
pages reserved      1000    1024
page extent gaps      64      67
```

To report faults found in a database object

sp_dbcc_faultreport reports faults in the specified database object that occurred on or before the specified date. You can generate a short or long report. The following example shows a short report:

```
sp_dbcc_faultreport ’short’
```

```
Database Name : sybsystemprocs

 Table Name     Index  Type Code Description        Page Number
 -------------- ------ --------- ------------------ -----------
 sysprocedures       0    100031 page not allocated        5702
 sysprocedures       1    100031 page not allocated       14151
 syslogs             0    100022 chain start error        24315
 syslogs             0    100031 page not allocated       24315
```

The following example shows part of the output of a long report for the sybsystemprocs database. The complete report repeats the information for each object in the target database.

```
sp_dbcc_faultreport ’long’
```

```
Generating 'Fault Report' for object sysprocedures in database sybsystemprocs.

Type Code: 100031; Soft fault, possibly spurious
Page reached by the chain is not allocated.
```

```
page id:  14151
page header:
0x00003747000037880000374600000005000648B803EF0001000103FE0080000F
Header for 14151, next 14216, previous 14150, id = 5:1
 time stamp = 0x0001000648B8, next row = 1007, level = 0
 free offset = 1022, minlen = 15, status = 128(0x0080)
.
```

To report statistics information from *dbcc_counter*

sp_dbcc_statisticsreport reports statistics information from the dbcc_counter table generated by dbcc checkstorage on or before the specified date. The following example shows output from this command:

```
sp_dbcc_statisticsreport 'sybsystemprocs',
'sysobjects'
```

```
Statistics Report on object sysobjects in database sybsystemprocs

 Parameter Name    Index Id Value
 ----------------- -------- -----------
 count             0        160.0
 max size          0        99.0
 max count         0        16.0
 bytes data        0        12829.0
 bytes used        0        15228.0
 count             1        16.0
 max size          1        9.0
 max level         1        0.0
 max count         1        16.0
 bytes data        1        64.0
 bytes used        1        176.0
 count             2        166.0
 max level         2        1.0
 max size          2        39.0
 max count         2        48.0
 bytes data        2        3092.0
 bytes used        2        4988.0


 Parameter Name        Index Id Partition Value    Dev_name
 --------------------- -------- --------- -------- ----------------
 page gaps             0        1         16.0     master
 pages used            0        1         17.0     master
 extents used          0        1         3.0      master
```

```
overflow pages         0        1          0.0      master
pages overhead         0        1          1.0      master
pages reserved         0        1          6.0      master
page extent gaps       0        1          7.0      master
ws buffer crosses      0        1          7.0      master
page extent crosses    0        1          7.0      master
page gaps              1        1          1.0      master
pages used             1        1          2.0      master
extents used           1        1          1.0      master
overflow pages         1        1          0.0      master
pages overhead         1        1          1.0      master
pages reserved         1        1          6.0      master
page extent gaps       1        1          0.0      master
ws buffer crosses      1        1          0.0      master
page extent crosses    1        1          0.0      master
page gaps              2        1          5.0      master
pages used             2        1          8.0      master
extents used           2        1          1.0      master
overflow pages         2        1          0.0      master
pages overhead         2        1          1.0      master
pages reserved         2        1          0.0      master
page extent gaps       2        1          0.0      master
ws buffer crosses      2        1          0.0      master
page extent crosses    2        1          0.0      master
```

Using *grant dbcc* and *revoke dbcc*

System Administrators can grant the execution of database consistency check (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.

Server-wide and database-specific *dbcc* commands

dbcc commands are either:

- Database-specific – dbcc commands such as checkalloc and checkstorage that execute on a particular target database. Although these commands are database-specific, only System Administrators can grant or revoke them.
- 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 grant dbcc command in all databases by making them valid users in the master database. However, it may be more convenient to allow 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 set a grant dbcc checkstorage command on all databases to a user-defined role called storage_admin_role, thereby eliminating the need to set grant dbcc checkstorage to storage_admin_role in every database.

Storing *grant* and *revoke* protection information

Protection information is stored in the sysprotects table at the database level. For database-specific dbcc commands that work only in a particular database, the information is stored in the sysprotects table of the target database.

grant and revoke commands can apply only to local objects in the database, except:

- create database
- connect
- set session authorization/set proxy

To grant permissions for these commands to a user, that user must be a valid user in master.

The following commands are 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:

- Server-wide grants – the login must be a valid user in master. Database-level commands granted at the server level do not give access to databases where the user is not a valid user. It only avoids having to grant permission in each database separately.
- Database-specific grants – the login should be a valid user in the target database.

See the *Reference Manual* for syntax and usage information about grant dbcc and revoke dbcc.

CHAPTER 27

Developing a Backup and Recovery Plan

Adaptive Server has **automatic recovery** procedures that protect you from power outages and computer failures. To protect yourself against media failure, you must make regular and frequent backups of your databases.

This chapter provides information to help you develop a backup and recovery plan. The first part of this chapter provides an overview of Adaptive Server's backup and recovery processes. The second part of this chapter discusses the backup and recovery issues that you should address before you begin using your system for production.

Keeping track of database changes

Adaptive Server uses transactions to keep track of all database changes. Transactions are Adaptive Server's units of work. A transaction consists of one or more Transact-SQL statements that succeed—or fail—as a unit.

Each SQL statement that modifies data is considered a **transaction**. Users can also define transactions by enclosing a series of statements within a begin transaction...end transaction block. For more information about transactions, see Chapter 18, "Transactions: Maintaining Data Consistency and Recovery," in the *Transact-SQL User's Guide*.

Each database has its own **transaction log**, the system table syslogs. The transaction log automatically records every transaction issued by each user of the database. You cannot turn off transaction logging.

The transaction log is a *write-ahead log*. When a user issues a statement that will modify the database, Adaptive Server writes the changes to the log. After all changes for a statement have been recorded in the log, they are written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page. At that time, it is written to disk.

If any statement in a transaction fails to complete, Adaptive Server reverses all changes made by the transaction. Adaptive Server writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction.

Getting information about the transaction log

The transaction log contains enough information about each transaction to ensure that it can be recovered. Use the dump transaction command to copy the information it contains to tape or disk. Use sp_spaceused syslogs to check the size of the log, or sp_helpsegment logsegment to check the space available for log growth.

Warning! Never use insert, update, or delete commands to modify syslogs.

Synchronizing a database and its log: checkpoints

A checkpoint writes all dirty pages—pages that have been modified in memory, but not on disk, since the last checkpoint—to the database device. Adaptive Server's automatic **checkpoint** mechanism guarantees that data pages changed by completed transactions are regularly written from the memory cache to the database device. Synchronizing the database and its transaction log shortens the time it takes to recover the database after a system crash.

Setting the recovery interval

Typically, automatic recovery takes from a few seconds to a few minutes per database. The time varies, depending on the size of the database, the size of the transaction log, and the number and size of the transactions that must be committed or rolled back.

Use sp_configure with the recovery interval in minutes parameter to specify, the maximum permissible recovery time. Adaptive Server runs automatic checkpoints often enough to recover the database within that period of time:

```
sp_configure "recovery interval in minutes"
```

The default value, 5, allows recovery within 5 minutes per database. To change the recovery interval to 3 minutes, use:

```
sp_configure "recovery interval in minutes", 3
```

Note The recovery interval has no effect on long-running, minimally logged transactions (such as create index) that are active at the time Adaptive Server fails. It may take as much time to reverse these transactions as it took to run them. To avoid lengthy delays, dump each database immediately after you create an index on one of its tables.

Automatic checkpoint procedure

Approximately once a minute, the checkpoint task checks each database on the server to see how many records have been added to the transaction log since the last checkpoint. If the server estimates that the time required to recover these transactions is greater than the database's recovery interval, Adaptive Server issues a checkpoint.

The modified pages are written from cache onto the database devices, and the checkpoint event is recorded in the transaction log. Then, the checkpoint task "sleeps" for another minute.

To see the checkpoint task, execute sp_who. The checkpoint task is usually displayed as "CHECKPOINT SLEEP" in the "cmd" column:

```
spid  status     loginame   hostname blk dbname  cmd
----- ---------- ---------- -------- --- ------- ----------------
    1 running    sa         mars     0   master  SELECT
    2 sleeping   NULL                0   master  NETWORK HANDLER
    3 sleeping   NULL                0   master  MIRROR HANDLER
    4 sleeping   NULL                0   master  HOUSEKEEPER
    5 sleeping   NULL                0   master  CHECKPOINT SLEEP
```

Checkpoint after user database upgrade

Adaptive Server inserts a checkpoint record immediately after upgrading a user database. Adaptive Server uses this record to ensure that a dump database occurs before a dump transaction occurs on the upgraded database.

Truncating the log after automatic checkpoints

System Administrators can truncate the transaction log when Adaptive Server performs an automatic checkpoint.

To set the trunc log on chkpt database option, which will truncate the transaction log if it consists of 50 or more rows when an automatic checkpoint occurs, execute this command from the master database:

```
sp_dboption database_name, "trunc log on chkpt", true
```

This option is not suitable for production environments because it does not make a copy of the transaction log before truncating it. Use trunc log on chkpt only for:

- Databases whose transaction logs cannot be backed up because they are not on a separate segment
- Test databases for which current backups are not important

Note If you leave the trunc log on chkpt option set to off (the default condition), the transaction log continues to grow until you truncate it with the dump transaction command.

To protect your log from running out of space, you should design your last-chance threshold procedure to dump the transaction log. For more information about threshold procedures, see Chapter 31, “Managing Free Space with Thresholds.”

Free checkpoints

When Adaptive Server has no user tasks to process, a housekeeper wash task automatically begins writing dirty buffers to disk. If the housekeeper task is able to flush all active buffer pools in all configured caches, it wakes up the checkpoint task. The checkpoint task determines whether it needs to perform a checkpoint on the database.

Checkpoints that occur as a result of the housekeeper wash task are known as *free checkpoints*. They do not involve writing many dirty pages to the database device, since the housekeeper wash task has already done this work. They may result in a shorter recovery time for the database.

For information about tuning the housekeeper task, see Chapter 4, “Using Engines and CPUs,” in *Performance and Tuning: Basics*.

Manually requesting a checkpoint

Database Owners can issue the checkpoint command to force all modified pages in memory to be written to disk. Manual checkpoints do not truncate the log, even if the trunc log on chkpt option of sp_dboption is turned on.

Use the checkpoint command:

- As a precautionary measure in special circumstances—for example, just before a planned shutdown with nowait so that Adaptive Server’s recovery mechanisms will occur within the recovery interval. (An ordinary shutdown performs a checkpoint.)

- To cause a change in database options to take effect after executing the sp_dboption system procedure. (After you run sp_dboption, an informational message reminds you to run checkpoint.)

You can use the checkpoint to identify the one or more databasess or use an all clause.

```
checkpoint [all | [dbname[, dbname[, dbname.....]]]
```

Automatic recovery after a system failure or shutdown

Each time you restart Adaptive Server—for example, after a power failure, an operating system failure, or the use of the shutdown command—it automatically performs a set of recovery procedures on each database.

The recovery mechanism compares each database to its transaction log. If the log record for a particular change is more recent than the data page, the recovery mechanism reapplies the change from the transaction log. If a transaction was ongoing at the time of the failure, the recovery mechanism reverses all changes that were made by the transaction.

When you start Adaptive Server, it performs database recovery in this order:

1. Recovers master
2. Recovers sybsystemprocs
3. Recovers model
4. Creates tempdb (by copying model)
5. Recovers sybsystemdb
6. Recovers sybsecurity
7. Recovers user databases, in order by sysdatabases.dbid, or according to the order specified by sp_dbrecovery_order. See below for more information about sp_dbrecovery_order.

Users can log in to Adaptive Server as soon as the system databases have been recovered, but they cannot access other databases until they have been recovered.

Determining whether messages are displayed during recovery

The configuration variable print recovery information determines whether Adaptive Server displays detailed messages about each transaction on the console screen during recovery. By default, these messages are not displayed. To display messages, use:

```
sp_configure "print recovery information", 1
```

Fast recovery

During a server restart after a planned or unplanned shutdown, or during HA failover, a significant portion of time is spent on database recovery. Faster recovery minimizes database downtime. The goal of fast recovery is to:

- Enhance the performance of database recovery.
- Recover multiple databases in parallel by making use of available server resources and tuning them intelligently.
- Provide multiple checkpoint tasks at runtime that can run concurrently to minimize the work at recovery time.

Adaptive Server start-up sequence

The following is the sequence of events at Adaptive Server start-up:

1. System databases are recovered on engine 0.
2. Adaptive Server accepts user connections.
3. All engines that are configured to be online during start-up are brought online.
4. User databases are recovered in parallel by a “self-tuned” number of recovery tasks using the default data cache tuned for optimal recovery performance.

   For more information about recovery-tuned parameters affecting the default data cache, see “Database recovery enhancements” on page 819. For more information about the number of recovery tasks, see “Parallel recovery” on page 818.

During an HA failover, failed over user databases are recovered and brought online in parallel. Fast recovery helps recovery from a planned or unplanned shutdowns, unplanned crashes, and HA failover.

Bringing engines online early

Engines are brought online after system databases are recovered, and before user databases. This allows user databases to be recovered in parallel, and makes the engines available for online activities.

Engines are brought online in this fashion during start-up only. In other circumstances, such as failover, engines are already online on the secondary server.

Parallel recovery

With Adaptive Server 12.5.1, during start-up and HA failover, databases are recovered in parallel by multiple recovery tasks. Database recovery is an I/O-intensive process. The time to recover Adaptive Server with parallel recovery depends on the bandwidth of the underlying I/O subsystem. The I/O subsystem should be able to handle Adaptive Server's concurrent I/O requests.

With parallel recovery, multiple tasks recover user databases concurrently. The number of recovery tasks is dependent on the configuration parameter max concurrently recovered db. The default value of 0 indicates that Adaptive Server adopts a self-tuning approach in which it does not make any assumptions on the underlying storage architecture. Statistical I/O sampling methods determine the optimal number of recovery tasks depending on the capabilities of the underlying I/O subsystem. An advisory on the optimal number of recovery tasks is provided. If the configuration value is non-zero, Adaptive Server spawns as many tasks as indicated by the configuration parameter.

During parallel recovery, the System Administrator can force serial recovery by reconfiguring the parameter to 1. The active recovery tasks drain out after completing the recovery of the database that is being worked on. The remaining databases are recovered serially.

max concurrently recovered db

max concurrently recovered db determines the degree of parallelism. The minimum value is 1, but you can also use the default value of 0, directing Adaptive Server to use a self-tuning apparoach. The maximum value is the number of engines at startup minus 1. max concurrently recovered db is also limited by the value of the configuration parameter number of open databases.

The default value is 0, which indicates automatic self-tuning by the server to determine the appropriate number of recovery tasks. A value of 1 indicates serial recovery.

Database recovery enhancements

Some of the database recovery enhancements made to Adaptive Server version 12.5.1 include:

- Log I/O size – Adaptive Server uses the largest buffer pool available in the default data cache for log I/O. If a pool with the largest buffer size is not available, the server creates this pool dynamically, and uses the pool for log I/O. The buffers for this pool come from the default pool. Recovery tunes the size of the large buffer pool for optimal recovery performance. Also, if the large pool is available but the size is not optimal, Adaptive Server dynamically resizes it and the default pool for optimal recovery performance. The buffer pool configurations are restored at the end of recovery.

  For more information on the largest buffer pool, pool sizes, and sp_poolconfig commands, see the *Performance and Tuning Guide*.

- async prefetch limit – during recovery, the server automatically sets the local async prefetch limit for the pools in the default data cache used by recovery to an optimal value. This overrides any user specifications for the duration of recovery.

  When recovery completes, the original configuration values are restored.

Recovery order

Since Adaptive Server 11.9.2, users have been able to specify the order in which databases are recovered for all or a subset of user databases. You can configure more important databases to be recovered earlier using sp_dbrecovery_order.

Adaptive Server 12.5.1 uses parallel recovery tasks to determine the next database to be recovered according to the user-specified order. The remaining databases are recovered in the order of their database IDs. The time to recover a database is dependent on many factors, including the size of the recoverable log. Hence, recovery can complete in an order other than which it started. For applications that need to enforce that databases are brought online in the same order as the recovery order, Adaptive Server provides the strict option in sp_dbrecovery_order.

sp_dbrecovery_order has an additional parameter indicating the online ordering.

```
sp_dbrecovery_order [database_name [, rec_order [,
force [ relax | strict ]]]]
```

- relax – the databases are made as they recover (default).
- strict – the databases are specified by the recovery order.

The default is relax, which means that databases are brought online immediately when recovery has completed.

Parallel checkpoints

A pool of checkpoint tasks works on the list of active databases in parallel. This pool is controlled by the configuration parameter number of checkpoint tasks. Where there is a checkpoint bottleneck, more checkpoint tasks translate to shorter recoverable logs, and recovery has less work to do in case of a crash, thus improving availability.

The default value of number of checkpoint tasks is 1, for serial checkpoints. The number of engines and number of open databases limit the value for this parameter. The absolute maximum value for this parameter is 8. This configuration parameter is dynamic. When the value for this parameter is reduced, checkpoints drain out, and when the value is increased, additional tasks are spawned.

The effectiveness of parallel checkpoints is dependent on the layout of the databases and performance of the underlying I/O subsystem, as checkpoints are I/O-intensive. This parameter should be tuned depending on the number of active databases and the ability of the I/O subsystem to handle writes.

number of checkpoint tasks

Use the configuration parameter number of checkpoint tasks, to configure parallel checkpoints. The maximum value is limited by the value of the configuration parameters number of engines online at startup and number of open databases, with an absolute ceiling of 8.

The default value is 1, which implies serial checkpoints is the default behavior. This parameter is also limited by the value of the configuration parameter number of open databases.

Recovery state

The global variable @@ *recovery_state* determines if Adaptive Server is in recovery. The values that this global variable can have are:

- NOT_IN_RECOVERY – Adaptive Server is not in start-up recovery or in failover recovery. Recovery has been completed and all databases that can be online are brought online.
- RECOVERY_TUNING – Adaptive Server is in recovery (either start-up or failover) and is tuning the optimal number of recovery tasks.
- BOOTIME_RECOVERY – Adaptive Server is in start-up recovery and has completed tuning the optimal number of tasks. Not all databases have been recovered.
- FAILOVER_RECOVERY – Adaptive Server is in recovery during an HA failover and has completed tuning the optimal number of recovery tasks. All databases are not brought online yet.

@@recovery_state can be used by applications to determine when all the databases are recovered and brought online.

Tuning for fast recovery

This section discusses are some guidelines on tuning Adaptive Server to reduce recovery time.

Database layout

- Databases should have logs and data on their own physical devices. The access patterns for log and data are different and should be kept separate.
- The underlying I/O subsystem should be configured to handle concurrent I/O requests from multiple databases in Adaptive Server.

Runtime configuration suggestions

- Configure an optimal number of checkpoint tasks using number of checkpoint tasks. Checkpointing is I/O intensive and should take into account the expected runtime performance and underlying I/O-subsystem performance. Tune number of checkpoint tasks according to the number of active databases and the ability of the I/O subsystem to handle writes. Tune the frequency of checkpoints using the configuration variable recovery interval in minutes.
- Configure an optimal housekeeper wash percentage controlled by housekeeper free write percent, so that during free cycles dirty pages are written out. The default value is usually optimal.
- Configure the order in which user databases should be recovered. The recovery order can be specified for all or a subset of user databases using sp_dbrecovery_order. For such databases, recovery is started in the order specified. Recovery is started in the database ID order, for databases that do not have a recovery order specified. If you need to bring databases online in the order in which they are recovered, use the strict option in sp_dbrecovery_order.
- Ensure that long-running transactions are kept to a minimum. Long-running transactions hold resources and can also cause longer recovery times.
- Shut down the server using polite shutdown to avoid longer recovery times.

Recovery time suggestions

- To facilitate parallel recovery, configure the maximum number of engines to be online at start-up.
- Configure an optimal number of recovery tasks using the configuration parameter max concurrently recovered db. Recovery is I/O-intensive and the value should take into account the response time of the underlying I/O subsystem. This value can be set to 0 (which is the default) for Adaptive Server to determine the optimal number of recovery tasks.
- As recovery uses only the default data cache, cache configuration should ensure that there is a large amount of default data cache for recovery.
- If data space accounting is not essential for a database, set the database option to turn off free space accounting using sp_dboption. This disables threshold actions on the data segment.

User-defined database recovery order

sp_dbrecovery_order allows you to determine the order in which individual user databases recover. This makes it possible to assign a recovery order in which, for example, critical databases recover before lower-priority databases.

Important features of recovery order are:

- System databases are recovered first, in this order:

  a master

  b sybsystemprocs

  c model

  d tempdb

  e sybsystemdb

  f sybsecurity

  All other databases are considered user databases, and you can specify their recovery order.

- You can use sp_dbrecovery_order to specify the recovery order of user-databases and to list the user-defined recovery order of an individual database or of all databases.
- User databases that are not explicitly assigned a recovery order with sp_dbrecovery_order are recovered according to their database ID, after all the databases that have a user-defined recovery order.
- If you do not use sp_dbrecovery_order to assign any databases a recovery order, user databases are recovered in order of database ID.

Using *sp_dbrecovery_order*

To use sp_dbrecovery_order to enter or modify a user-defined recovery order, you must be in the master database and have System Administrator privileges. Any user, in any database, can use sp_dbrecovery_order to list the user-defined recovery order of databases.

The syntax for sp_dbrecovery_order is:

sp_dbrecovery_order
    [*database_name* [, *rec_order* [, force]]]

where:

- *database_name* – is the name of the user database to which you want to assign a recovery order
- *rec_order* – is the order in which the database is to be recovered

Recovery order must be consecutive, starting with 1. You cannot assign a recovery sequence of 1, 2, 4, with the intention of assigning a recovery order of 3 to another database at a later time.

To insert a database into a user-defined recovery sequence without putting it at the end, enter *rec_order* and specify force. For example, if databases A, B, and C have a user-defined recovery order of 1, 2, 3, and you want to insert the pubs2 database as the second user database to recover, enter:

```
sp_dbrecovery_order pubs2, 2, force
```

This command assigns a recovery order of 3 to database B and a recovery order of 4 to database C.

Changing or deleting the recovery position of a database

To change the position of a database in a user-defined recovery sequence, delete the database from the recovery sequence and then insert it in the position you want it to occupy. If the new position is not at the end of the recovery order, use the force option.

To delete a database from a recovery sequence, specify a recovery order of -1.

For example, to move the pubs2 database from recovery position 2 to recovery position 1, delete the database from the recovery sequence and then reassign it a recovery order as follows:

```
sp_dbrecovery_order pubs2, -1
sp_dbrecovery_order pubs2, 1, "force"
```

Listing the user-assigned recovery order of databases

To list the recovery order of all databases assigned a recovery order, use:

```
sp_dbrecovery_order
```

This generates output similar to:

```
The following databases have user specified recovery order:
Recovery Order Database Name        Database Id
-------------- -------------------- -----------
    1          dbccdb                   8
    2          pubs2                    5
    3          pubs3                    6
    4          pubtune                  7
The rest of the databases will be recovered in default database id order.
```

To display the recovery order of a specific database, enter the database name:

```
1> sp_dbrecovery_order pubs2
2> go

Database Name Database id Recovery Order
 ------------- ----------- --------------
 pubs2             5          2
```

Fault isolation during recovery

The recovery procedures, known simply as "recovery," rebuild the server's databases from the transaction logs. The following situations cause recovery to run:

- Adaptive Server start-up
- Use of the load database command
- Use of the load transaction command

The recovery isolation mode setting controls how recovery behaves when it detects corrupt data while reversing or reapplying a transaction in a database.

If an index is marked as suspect, the System Administrator can repair this by dropping and re-creating the index.

Recovery fault isolation provides the ability to:

- Configure whether an entire database or just the suspect pages become inaccessible when recovery detects corruption
- Configure whether an entire database with suspect pages comes online in read_only mode or whether the online pages are accessible for modification
- List databases that have suspect pages
- List the suspect pages in a specified database by page ID, index ID, and object name
- Bring suspect pages online for the System Administrator while they are being repaired
- Bring suspect pages online for all database users after they have been repaired

The ability to isolate only the suspect pages while bringing the rest of the database online provides a greater degree of flexibility in dealing with data corruption. You can diagnose problems, and sometimes correct them, while most of the database is accessible to users. You can assess the extent of the damage and schedule emergency repairs or reload for a convenient time.

Recovery fault isolation applies only to user databases. Recovery always takes a system database entirely offline if it has any corrupt pages. You cannot recover a system database until you have repaired or removed all of its corrupt pages.

Persistence of offline pages

Suspect pages that you have taken offline remain offline when you restart the server. Information about offline pages is stored in master.dbo.sysattributes.

Use the drop database and load database commands to clear entries for suspect pages from master.dbo.sysattributes.

Configuring recovery fault isolation

When Adaptive Server is installed, the default recovery isolation mode is “databases,” which marks a database as suspect and takes the entire database offline if it detects any corrupt pages.

Isolating suspect pages

To isolate the suspect pages so that only they are taken offline, while the rest of the database remains accessible to users, use the sp_setsuspect_granularity to set the recovery isolation mode to “page.” This mode will be in effect the next time that recovery is performed in the database.

The syntax for sp_setsuspect_granularity is:

```
sp_setsuspect_granularity
    [dbname [,{"database" | "page"} [, "read_only"]]]
```

With the *dbname* and either database or page as the second argument, sp_setsuspect_granularity sets the recovery isolation mode.

Without the database or page argument, sp_setsuspect_granularity displays the current and configured recovery isolation mode settings for the specified database. Without any arguments, it displays those settings for the current database.

If corruption cannot be isolated to a specific page, recovery marks the entire database as suspect, even if you set the recovery isolation mode to “page.” For example, a corrupt transaction log or the unavailability of a global resource causes this to occur.

When recovery marks specific pages as suspect, the default behavior is for the database to be accessible for reading and writing with the suspect pages offline and therefore inaccessible. However, if you specify the read_only option to sp_setsuspect_granularity, and recovery marks any pages as suspect, the entire database comes online in read_only mode and cannot be modified. If you prefer the read_only option, but in certain cases you are comfortable allowing users to modify the non-suspect pages, you can make the online portion of the database writable with sp_dboption:

```
sp_dboption pubs2, "read only", false
```

In this case, the suspect pages remain offline until you repair them or force them, as described in "Bringing offline pages online" on page 830.

Raising the number of suspect pages allowed

The suspect escalation threshold is the number of suspect pages at which recovery marks an entire database suspect, even if the recovery isolation mode is "page." By default, it is set to 20 pages in a single database. You can use sp_setsuspect_threshold to change the suspect escalation threshold.

The syntax for sp_setsuspect_threshold is:

sp_setsuspect_threshold [*dbname* [,*threshold*]]

With the *dbname* and *threshold* arguments, sp_setsuspect_threshold displays the current and configured suspect escalation threshold settings for the specified database. Without any arguments, it displays these settings for the current database.

You configure recovery fault isolation and the suspect escalation threshold at the database level.

You cannot execute sp_setsuspect_granularity or sp_setsuspect_threshold inside a transaction.

You must have the sa_role and be in the master database to set values with sp_setsuspect_granularity and sp_setsuspect_threshold. Any user can execute these procedures with only the name of the database as an argument to display the values configured for that database, as illustrated below:

```
sp_setsuspect_granularity pubs2

DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  ------------------  ------------------  -----------
pubs2    page                page                read/write
```

```
sp_setsuspect_threshold pubs2

DB Name       Cur. Suspect threshold   Cfg. Suspect threshold
------------- ------------------------ ----------------------
pubs2         20                       30
```

This example shows that the recovery isolation mode for the pubs2 database was "page" and the escalation threshold was 20 the last time recovery ran on this database (the current suspect threshold values). The next time recovery runs on this database, the recovery isolation mode will be "page" and the escalation threshold will be 30 (the configured values).

With no arguments, sp_setsuspect_granularity and sp_setsuspect_threshold display the current and configured settings for the current database, if it is a user database.

Getting information about offline databases and pages

To see which databases have offline pages, use sp_listsuspect_db. The syntax is:

sp_listsuspect_db

The following example displays general information about the suspect pages:

```
sp_listsuspect_db

The database 'dbt1' has 3 suspect pages belonging to 2 objects.
(return status = 0)
```

To get detailed information about the individual offline pages, use sp_listsuspect_page. The syntax is:

sp_listsuspect_page [*dbname*]

If you don't specify the dbname, the defaults is the current database. The following example shows the detailed page-level output of sp_listsuspect_page in the dbt1 database.

```
sp_listsuspect_page dbt1

DBName   Pageid       Object    Index   Access
-------- ------------ --------- ------- -------------------
dbt1     384          tab1      0       BLOCK_ALL
dbt1     390          tab1      0       BLOCK_ALL
```

```
dbt1     416          tab1      1       SA_ONLY

(3 rows affected, return status = 0)
```

If the value in the “Access” column is SA_ONLY, the suspect page is 1, the suspect page is accessible to users with the sa_role. If it is BLOCK_ALL, no one can access the page.

Any user can run sp_listsuspect_db and sp_listsuspect_page from any database.

Bringing offline pages online

To make all the offline pages in a database accessible, use sp_forceonline_db. The syntax is:

```
sp_forceonline_db dbname,
   {"sa_on" | "sa_off" | "all_users"}
```

To make an individual offline page accessible, use sp_forceonline_page. The syntax is:

```
sp_forceonline_page dbname, pgid
   {"sa_on" | "sa_off" | "all_users"}
```

With both of these procedures, you specify the type of access.

- “sa_on” makes the suspect page or database accessible only to users with the sa_role. This is useful for repairing the suspect pages and testing the repairs while the database is up and running, without allowing normal users access to the suspect pages. You can also use it to perform a dump database or a dump transaction with no_log on a database with suspect pages, which would be prohibited if the pages were offline.
- “sa_off” blocks access to all users, including System Administrators. This reverses a previous sp_forceonline_db or sp_forceonline_page with “sa_on.”
- “all_users” brings offline pages online for all users after the pages have been repaired.

Unlike bringing suspect pages online with "sa_on" and then making them offline again with "sa_off," when you use sp_forceonline_page or sp_forceonline_db to bring pages online for "all users," this action cannot be reversed. There is no way to make the online pages offline again.

Warning! Adaptive Server does not perform any checks on pages being brought online. It is your responsibility to ensure that pages being brought online have been repaired.

You cannot execute sp_forceonline_db or sp_forceonline_page inside a transaction.

You must have the sa_role and be in the master database to execute sp_forceonline_db and sp_forceonline_page.

Index-level fault isolation for data-only-locked tables

When pages of an index for a data-only-locked table are marked as suspect during recovery, the entire index is taken offline. Two system procedures manage offline indexes:

- sp_listsuspect_object
- sp_forceonline_object

In most cases, a System Administrator uses sp_forceonline_object to make a suspect index available only to those with the sa_role. If the index is on a user table, you can repair the suspect index by dropping and re-creating the index.

See the *Reference Manual* for more information about sp_listsuspect_objec and sp_forceonline_object.

Side effects of offline pages

The following restrictions apply to databases with offline pages:

- Transactions that need offline data, either directly or indirectly (for example, because of referential integrity constraints), fail and generate a message.

- You cannot use dump database when any part of the database is offline.

  A System Administrator can force the offline pages online using sp_forceonline_db with “sa_on” dump the database, and then use sp_forceonline_db with “sa_off” after the dump completes.

- You cannot use dump transaction with no_log or dump transaction with truncate_only if any part of a database is offline.

  A System Administrator can force the offline pages online using sp_forceonline_db with “sa_on”, dump the transaction log using with no_log, and then use sp_forceonline_db with “sa_off” after the dump completes.

- If you want to drop a table or index containing offline pages, you must use a transaction in the master database. Otherwise, the drop will fail because it needs to delete entries for the suspect pages from master.dbo.sysattributes. The following example drops the object and deletes information about its offline pages from master.dbo.sysattributes.

  To drop an index named authors_au_id_ind, which contains suspect pages, from the pubs2 database, drop the index inside a master database transaction as follows:

  ```
  use master
  go
  sp_dboption pubs2, "ddl in tran", true
  go
  checkpoint pubs2
  go
  begin transaction
  drop index authors.au_id_ind
  commit
  go
  use master
  go
  sp_dboption pubs2, "ddl in tran", false
  go
  checkpoint pubs2
  go
  ```

Recovery strategies using recovery fault isolation

There are two major strategies for returning a database with suspect pages to a consistent state while users are accessing it: reload and repair.

Both strategies require:

- A clean database dump
- A series of reliable transaction log dumps up to the point at which the database is recovered with suspect pages
- A transaction log dump to a device immediately after the database is recovered to capture changes to the offline pages
- Continuous transaction log dumps to devices while users work in the partially offline database

Reload strategy

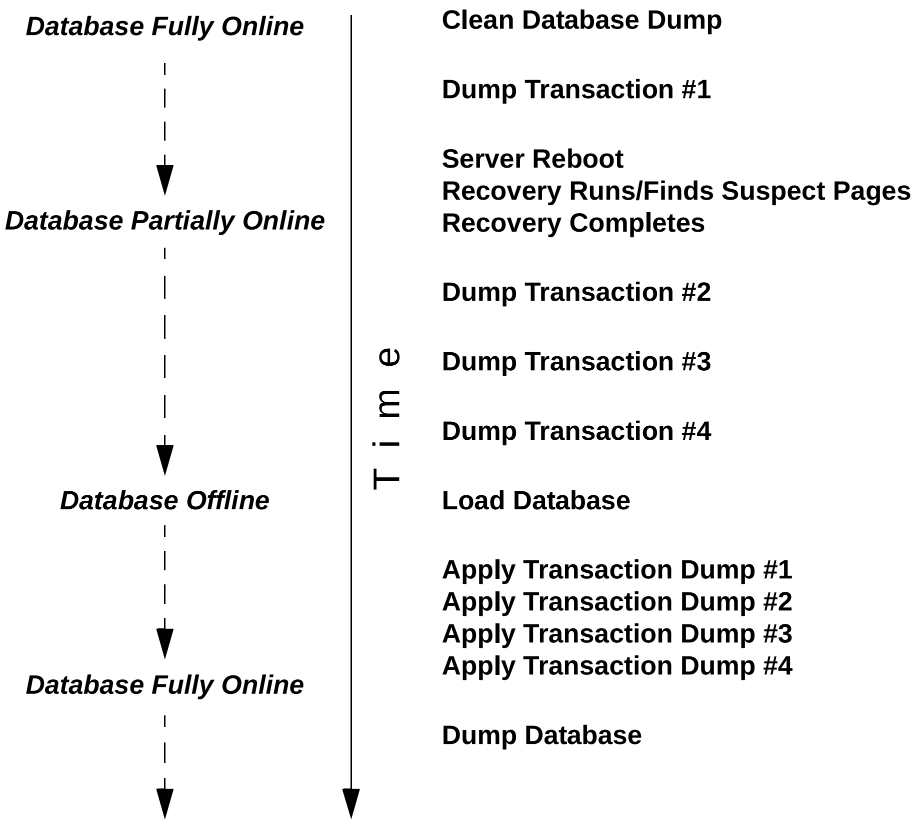
Reloading involves restoring a clean database from backups. When convenient, load the most recent clean database dump, and apply the transaction logs to restore the database.

load database clears the suspect page information from the master.dbo.sysdatabases and master.dbo.sysattributes system tables.

When the restored database is online, dump the database immediately.

Figure 27-1 illustrates the strategy used to reload databases.

Figure 27-1: Reload strategy



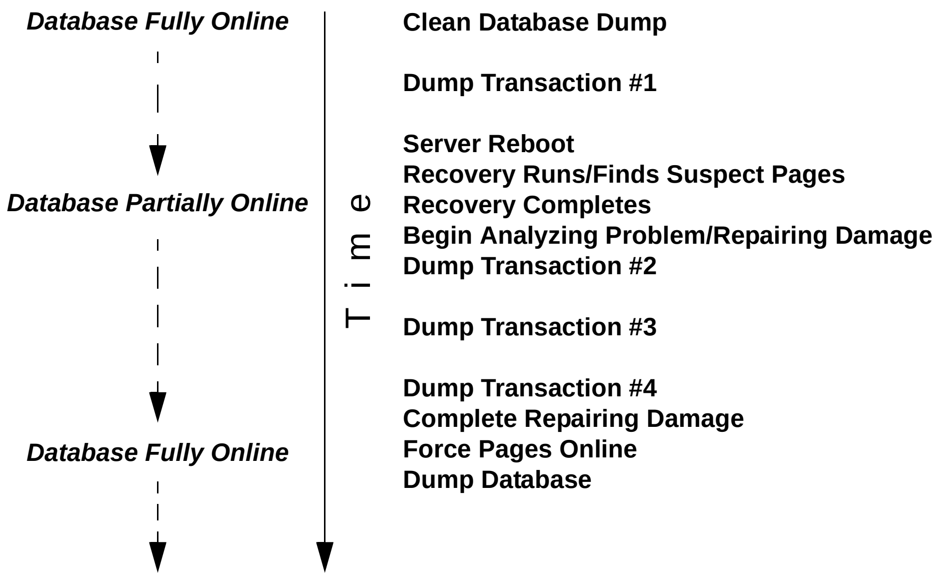
Repair strategy

The repair strategy involves repairing the corrupt pages while the database is partially offline. You diagnose and repair problems using known methods, including dbcc commands, running queries with known results against the suspect pages, and calling Sybase Technical Support, if necessary. Repairing damage can also include dropping and re-creating objects that contain suspect pages.

You can either use sp_forceonline_page to bring offline pages online individually, as they are repaired, or wait until all the offline pages are repaired and bring them online all at once with sp_forceonline_db.

The repair strategy does not require taking the entire database offline. Figure 27-2 illustrates the strategy used to repair corrupt pages.

Figure 27-2: Repair strategy



Assessing the extent of corruption

You can sometimes use recovery fault isolation to assess the extent of corruption by forcing recovery to run and examining the number of pages marked suspect and the objects to which they belong.

For example, if users report problems in a particular database, set the recovery isolation mode to “page,” and force recovery by restarting Adaptive Server. When recovery completes, use sp_listsuspect_db or sp_listsuspect_page to determine how many pages are suspect and which database objects are affected.

If the entire database is marked suspect and you receive the message:

```
Reached suspect threshold ’%d’ for database ’%.*s’.
Increase suspect threshold using
sp_setsuspect_threshold.
```

use sp_setsuspect_threshold to raise the suspect escalation threshold and force recovery to run again. Each time you get this message, you can raise the threshold and run recovery until the database comes online. If you do not get this message, the corruption is not isolated to specific pages, in which case this strategy for determining the number of suspect pages will not work.

Using the dump and load commands

In case of media failure, such as a disk crash, you can restore your databases if—and only if—you have regular backups of the databases and their transaction logs. Full recovery depends on the regular use of the dump database and dump transaction commands to back up databases and the load database and load transaction commands to restore them. These commands are described briefly below and more fully in Chapter 28, "Backing Up and Restoring User Databases," and Chapter 29, "Restoring the System Databases."

Warning! Never use operating system copy commands to copy a database device. Loading the copy into Adaptive Server causes massive database corruption.

The dump commands can complete successfully even if your database is corrupt. Before you back up a database, use the dbcc commands to check its consistency. See Chapter 26, "Checking Database Consistency," for more information.

Warning! If you dump directly to tape, do not store any other types of files (UNIX backups, tar files, and so on) on that tape. Doing so can invalidate the Sybase dump files. However, if you dump to a UNIX file system, the resulting files can be archived to a tape.

Making routine database dumps: *dump database*

The dump database command makes a copy of the entire database, including both the data and the transaction log. dump database does *not* truncate the log.

dump database allows **dynamic dumps**. Users can continue to make changes to the database while the dump takes place. This makes it convenient to back up databases on a regular basis.

dump database executes in three phases. A progress message informs you when each phase completes. When the dump is finished, it reflects all changes that were made during its execution, except for those initiated during phase 3.

Making routine transaction log dumps: *dump transaction*

Use the dump transaction command (or its abbreviation, dump tran) to make routine backups of your transaction log. dump transaction is similar to the incremental backups provided by many operating systems. It copies the transaction log, providing a record of any database changes made since the last transaction log dump. After dump transaction has copied the log, it truncates the inactive portion.

dump transaction takes less time and storage space than a full database backup, and it is usually run more often. Users can continue to make changes to the database while the dump is taking place. You can run dump transaction only if the database stores its log on a separate segment.

After a media failure, use the with no_truncate option of dump transaction to back up your transaction log. This provides a record of the transaction log up to the time of the failure.

Copying the log after device failure: *dump tran with no_truncate*

If your data device fails and the database is inaccessible, use the with no_truncate option of dump transaction to get a current copy of the log. This option does not truncate the log. You can use it only if the transaction log is on a separate segment and the master database is accessible.

Restoring the entire database: *load database*

Use the load database command to load the backup created with dump database. You can load the dump into a preexisting database or create a new database with the for load option. When you create a new database, allocate at least as much space as was allocated to the original database.

Warning! You cannot load a dump that was made on a different platform or generated on pre-version 10.0 SQL Server. If the database you are loading includes tables that contain the primary keys for tables in other databases, you must load the dump into a database with the same database name as the one dumped.

The load database command sets the database status to “offline.” This means you do not have to use the no chkpt on recovery, dbo use only, and read only options of sp_dboption before you load a database. However, no one can use a database during the database load and subsequent transaction log loads. To make the database accessible to users, issue the online database command.

After the database is loaded, Adaptive Server may need to:

- “Zero” all unused pages, if the database being loaded into is larger than the dumped database.
- Complete recovery, applying transaction log changes to the data.

Depending on the number of unallocated pages or long transactions, this can take a few seconds or many hours for a very large database. Adaptive Server issues messages that it is “zero-ing” pages or has begun recovery. These messages are normally buffered; to see them, issue:

```
set flushmessage on
```

Applying changes to the database: *load transaction*

After you have loaded the database, use the load transaction command (or its abbreviation, load tran) to load each transaction log dump *in the order in which it was made*. This process reconstructs the database by re-executing the changes recorded in the transaction log. If necessary, you can recover a database by rolling it forward to a particular time in its transaction log, using the until_time option of load transaction.

Users cannot make changes to the database between the load database and load transaction commands, due to the “offline” status set by load database.

You can load only the transaction log dumps that are at the same release level as the associated database.

When the entire sequence of transaction log dumps has been loaded, the database reflects all transactions that had committed at the time of the last transaction log dump.

Making the database available to users: *online database*

When the load sequence completes, change the database status to “online,” to make it available to users. A database loaded by load database remains inaccessible until you issue the online database command is issued.

Before you issue this command, be sure you have loaded all required transaction logs.

Moving a database to another Adaptive Server

You can use dump database and load database to move a database from one Adaptive Server to another, as long as both Adaptive Servers run on the same hardware and software platform. However, you must ensure that the device allocations on the target Adaptive Server match those on the original. Otherwise, system and user-defined segments in the new database will not match those in the original database.

To preserve device allocations when loading a database dump into a new Adaptive Server, use the same instructions as for recovering a user database from a failed device. See “Examining the space usage” on page 932 for more information.

Also, follow these general guidelines when moving system databases to different devices:

- Before moving the master database, always unmirror the master device. If you do not, Adaptive Server will try to use the old mirror device file when you start Adaptive Server with the new device.
- When moving the master database, use a new device that is the same size as the original to avoid allocation errors in sysdevices.

- To move the sybsecurity database, place the new database in single-user mode before loading the old data into it.

Upgrading a user database

You can load dumps into the current version of Adaptive Server from any version of Adaptive Server that is at version 10.0 and later. The loaded database is not upgraded until you issue online database.

The steps for upgrading user databases are the same as for system databases:

1. Use load database to load a database dump of a release 10.0 or later Adaptive Server. load database sets the database status to “offline.”
2. Use load transaction to load, *in order*, all transaction logs generated after the last database dump. Be sure you have loaded all transaction logs before going to step 3.
3. Use online database to upgrade the database. The online database command upgrades the database because its present state is incompatible with the current version of Adaptive Server. When the upgrade completes, the database status is set to “online,” which makes the database available for public use.
4. Make a dump of the upgraded database. A dump database must occur before a dump transaction command is permitted.

For more information about load database, load transaction, and online database, see the *Reference Manual*.

Using the special *dump transaction* options

In certain circumstances, the simple model described above does not apply. Table 27-1 describes when to use the special with no_log and with truncate_only options instead of the standard dump transaction command.

Warning! Use the special dump transaction commands *only* as indicated in Table 27-1. In particular, use dump transaction with no_log as a last resort and use it only once after dump transaction with no_truncate fails. The dump transaction with no_log command frees very little space in the transaction log. If you continue to load data after entering dump transaction with no_log, the log may fill completely, causing any further dump transaction commands to fail. Use the alter database command to allocate additional space to the database.

Table 27-1: When to use dump transaction with truncate_only or with no_log

When	Use
The log is on the same segment as the data.	dump transaction with truncate_only to truncate the log dump database to copy the entire database, including the log
You are not concerned with the recovery of recent transactions (for example, in an early development environment).	dump transaction with truncate_only to truncate the log dump database to copy the entire database
Your usual method of dumping the transaction log (either the standard dump transaction command or dump transaction with truncate_only) fails because of insufficient log space.	dump transaction with no_log to truncate the log without recording the event dump database immediately afterward to copy the entire database, including the log

Using the special load options to identify dump files

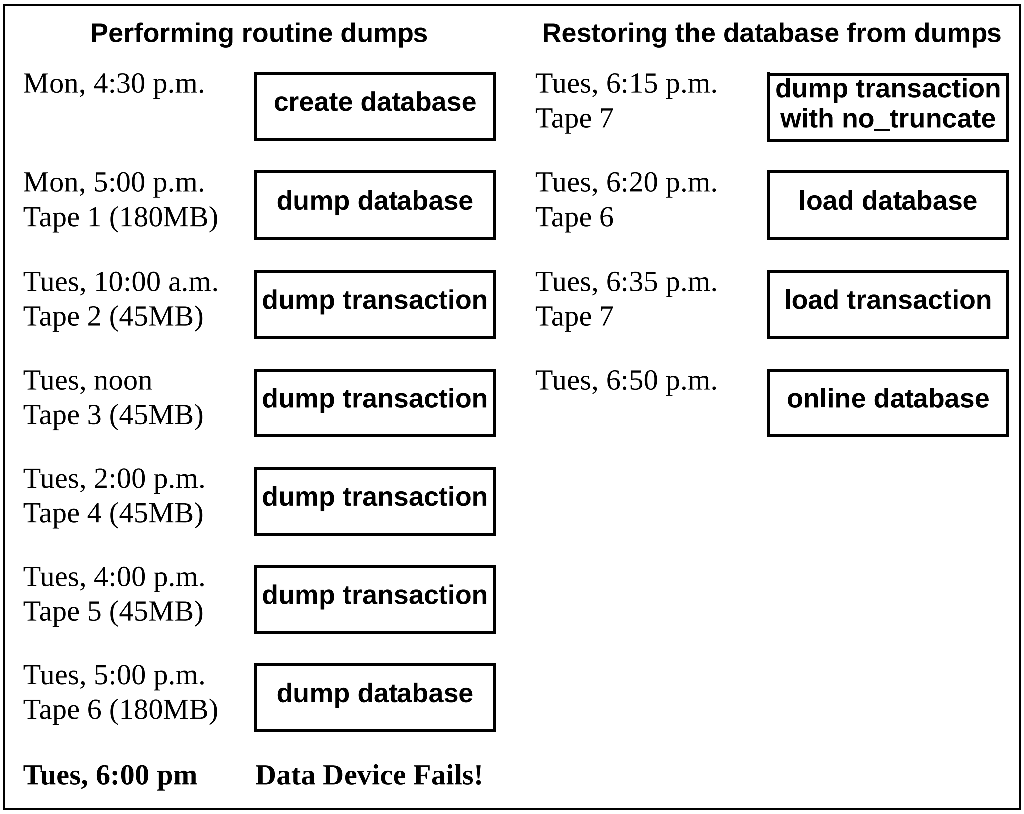
Use the with headeronly option to provide header information for a specified file or for the first file on a tape. Use the with listonly option to return information about all files on a tape. These options do not actually load databases or transaction logs on the tape.

Note These options are mutually exclusive. If you specify both, with listonly prevails.

Restoring a database from backups

Figure 27-3 illustrates the process of restoring a database that is created at 4:30 p.m. on Monday and dumped immediately afterward. Full database dumps are made every night at 5:00 p.m. Transaction log dumps are made at 10:00 a.m., 12:00 p.m., 2:00 p.m., and 4:00 p.m. every day:

Figure 27-3: Restoring a database, a scenario



Performing routine dumps		Restoring the database from dumps	
Mon, 4:30 p.m.	**create database**	Tues, 6:15 p.m. Tape 7	**dump transaction with no_truncate**
Mon, 5:00 p.m. Tape 1 (180MB)	**dump database**	Tues, 6:20 p.m. Tape 6	**load database**
Tues, 10:00 a.m. Tape 2 (45MB)	**dump transaction**	Tues, 6:35 p.m. Tape 7	**load transaction**
Tues, noon Tape 3 (45MB)	**dump transaction**	Tues, 6:50 p.m.	**online database**
Tues, 2:00 p.m. Tape 4 (45MB)	**dump transaction**		
Tues, 4:00 p.m. Tape 5 (45MB)	**dump transaction**		
Tues, 5:00 p.m. Tape 6 (180MB)	**dump database**		
Tues, 6:00 pm	**Data Device Fails!**		

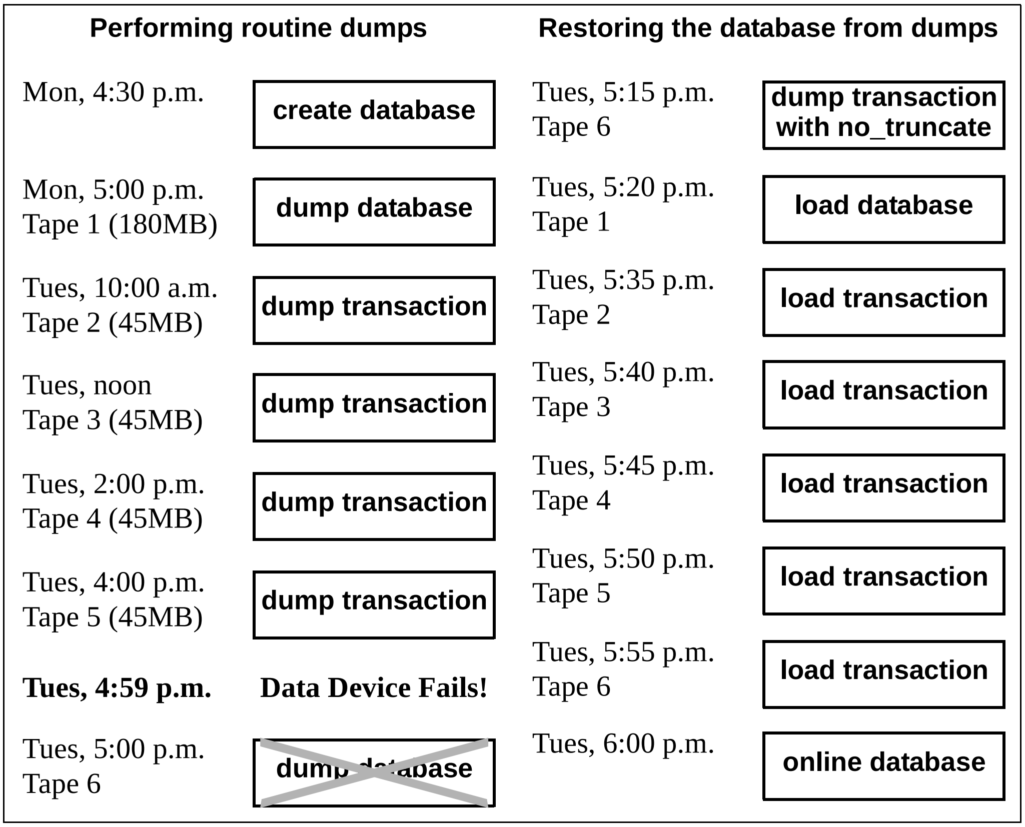
If the disk that stores the data fails on Tuesday at 6:00 p.m., follow these steps to restore the database:

1. Use dump transaction with no_truncate to get a current transaction log dump.
2. Use load database to load the most recent database dump, Tape 6. load database sets the database status to “offline.”
3. Use load transaction to apply the most recent transaction log dump, Tape 7.

4 Use online database to set the database status to “online.”

Figure 27-4 illustrates how to restore the database when the data device fails at 4:59 p.m. on Tuesday—just before the operator is scheduled to make the nightly database dump:

Figure 27-4: Restoring a database, a second scenario



Use the following steps to restore the database:

1 Use dump transaction with no_truncate to get a current transaction log dump on Tape 6 (the tape you would have used for the routine database dump).

2 Use load database to load the most recent database dump, Tape 1. load database sets the database status to “offline.”

3 Use load transaction to load Tapes 2, 3, 4, and 5 and the most recent transaction log dump, Tape 6.

4 Use online database to set the database status to “online.”

Suspending and resuming updates to databases

quiesce database hold allows you to block updates to one or more databases while you perform a disk unmirroring or external copy of each database device. Because no writes are performed during this time, the external copy (the secondary image) of the database is identical to the primary image. While the database is in the quiescent state, read-only queries to operations on the database are allowed. To resume updates to the database, issue quiesce database release when the external copy operation has completed. You can load the external copy of the database onto a secondary server, ensuring that you have a transactionally consistent copy of your primary image. You can issue quiesce database hold from one isql connection and then log in with another isql connection and issue quiesce database release.

The syntax for quiesce database is:

```
quiesce database tag_name hold database_name
    [, database_name]
[for external dump]
```

or,

```
quiesce database tag_name release
```

where:

- *tag_name* – is a user-defined label for the list of databases to hold or release
- *database_name* – is the name of a database for which you are suspending updates

---

Note *tag_name* must conform to the rules for identifiers. See the *Reference Manual* for a list of these rules. You must use the same *tag_name* for both quiesce database...hold and quiesce database...release.

For example, to suspend updates to the pubs2 database, enter:

```
quiesce database pubs_tag hold pubs2
```

Adaptive Server writes messages similar to the following to the error log:

```
QUIESCE DATABASE command with tag pubs_tag is being executed by process 9.

Process 9 successfully executed QUIESCE DATABASE with HOLD option for tag
pubs_tag. Processes trying to issue IO operation on the quiesced database(s)
will be suspended until user executes Quiesce Database command with RELEASE
option.
```

Any updates to the pubs2 database are delayed until the database is released, at which time the updates complete. To release the pubs2 database, enter:

```
quiesce database pubs_tag release
```

After releasing the database, you can bring up the secondary server with the -q parameter if you used the for external dump clause. Recovery makes the databases transactionally consistent, or you can wait to bring the database online and then apply the transaction log.

Guidelines for using quiesce database

The simplest way to use quiesce database is to make a full copy of an entire installation. This ensures that system mappings are consistent. These mappings are carried to the secondary installation when the system databases, that contain them are physically copied as part of quiesce database hold's set of databases. These mappings are fulfilled when all user databases in the source installation are copied as part of the same set. quiesce database allows for eight database names during a single operation. If a source installation has more than eight databases, you can issue multiple instances of quiesce database hold to create multiple concurrent quiescent states for multiple sets of databases.

To create the source installation from scratch, you can use almost identical scripts to create both the primary and secondary installations. The script for the secondary installation might vary in the physical device names passed to the disk init command. This approach requires that updates to system devices on the primary server be reflected by identical changes to the secondary server. For example, if you perform an alter database command on the primary server, you must also perform the same command on the secondary server using identical parameters. This approach requires that the database devices be supported by a volume manager, which can present to both the primary and secondary servers the same physical device names for devices that are physically distinct and separate.

Your site may develop its own procedures for making external copies of database devices. However, Sybase recommends the following:

- You include the master database in quiesce database's list of databases.
- You name devices using identical strings on both the primary and secondary servers.
- You make the environments for the master, model, and sybsystemprocs system databases in the primary and secondary installations identical. In particular, sysusages mappings and database IDs for the copied databases must be identical on the primary and secondary servers, and database IDs for both servers must be reflected identically in sysdatabases.
- The mapping between syslogins.suid and sysusers.suid must remain consistent in the secondary server.
- If the primary server and the secondary server share a copy of master, and if the sysdevices entry for each copied device uses identical strings, the *physname* values in both servers must be physically distinct and separate.
- Making external copies of a database has the following restrictions:
  - The copy process can begin only after quiesce database hold has completed.
  - Every device for every database in quiesce database's list of databases must be copied.
  - The external copy must finish before you invoke quiesce database release.
- During the interval that quiesce database provides for the external copy operation, updates are prevented on any disk space belonging to any database in quiesce database's list of databases. This space is defined in sysusages. However, if space on a device is shared between a database in quiesce database's list of databases and a database not in the list, updates to the shared device may occur while the external copy is made. When you are deciding where to locate databases in a system in which you plan to make external copies, you can either:
  - Segregate databases so they do not share devices in an environment where you will use quiesce database, or

  - Plan to copy all the databases on the device (this follows the recommendation above that you make a copy of the entire installation).

- Use quiesce database only when there is little update activity on the databases (preferably during a moment of read-only activity). When you quiesce the database during a quiet time, not only are fewer users inconvenienced, but, depending on the third-party I/O subsystem that is to perform the external copy, there may also be less time spent synchronizing devices involved in the copy operation.

- The mount and unmount commands make it easier to move or copy databases. You can move or copy a database from one Adaptive Server to another without rebooting the server. You can move or copy more than one database at a time using the mount and unmount commands.

  These commands can also be used when you need to physically move the devices and then reactivate the databases.

  When you unmount a database, you remove the database and its devices from an Adaptive Server. The unmount shuts down the database and drops it from the Adaptive Server, the devices are also deactivated and dropped. No changes are made to the database or its pages when unmounted.

Maintaining server roles in a primary and secondary relationship

If your site consists of two Adaptive Servers, one functioning as the primary server, and the other acting as a secondary server that receives external copies of the primary server’s databases, you must never mix the roles of these servers. That is, the role each server plays can change (the primary server can become the secondary server and vice-versa), but these roles cannot be fulfilled by the same server simultaneously.

Starting the secondary server with the *-q* option

The dataserver -q option identifies the secondary server. Do not use the -q option to start the primary server. Under the -q option, user databases that were copied during quiesce database for external dump stay offline until:

- You dump the transaction log for a database on the primary server with standby access (that is, dump tran with standby_access) followed by load tran to the copy of this database on the secondary server, and then perform online database for standby access on this database.
- You force the database online for read and write access by issuing online database. However, if you do this, the database recovery writes compensation log records, and you cannot load the transaction log without either loading the database, or making a new copy of the primary devices using quiesce database.

System databases come online regardless of the -q option, and write compensation log records for any transactions that are rolled back.

"in quiesce" database log record value updated

If you start the secondary server using the -q option of dataserver, for each user database marked internally as "in quiesce", Adaptive Server issues a message at start-up stating that the database is `"in quiesce."`

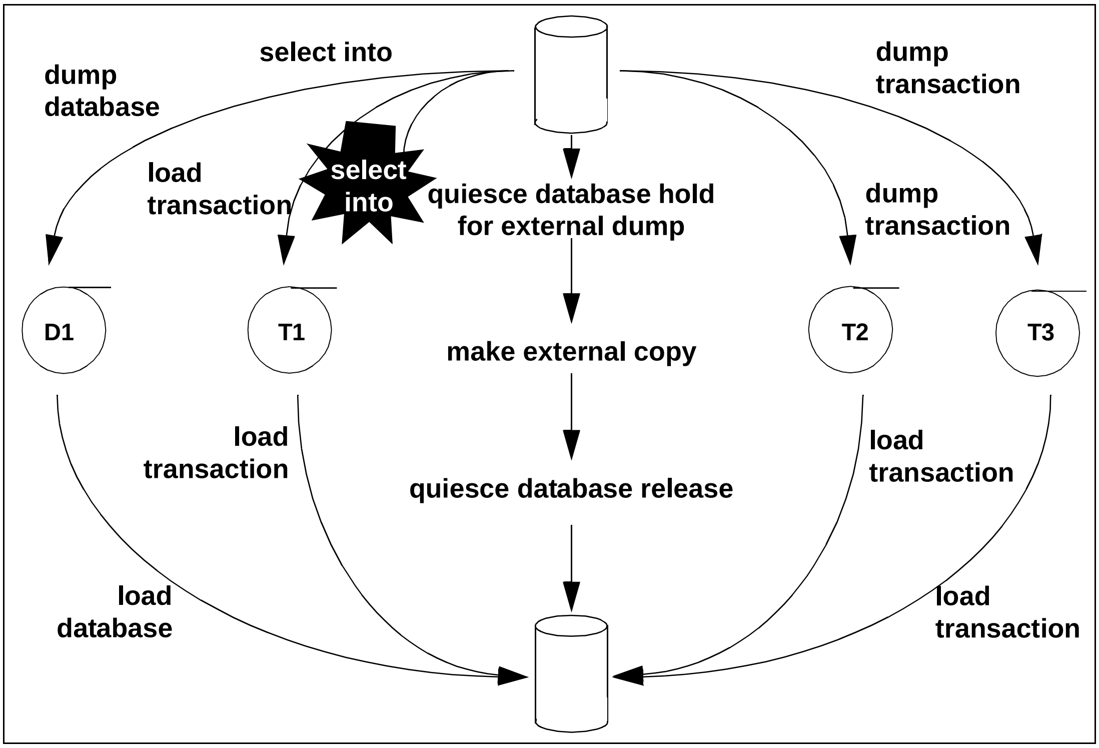
-q recovery for databases copied with quiesce database for external dump acts much like the recovery for load database. Like recovery for load database, it internally records the address of the current last log record, so that a subsequent load transaction can compare this address to the address of the previous current last log record. If these two values do not match, then there has been original activity in the secondary database, and Adaptive Server raises error number 4306.

Updating the dump sequence number

Like dump database, quiesce database updates the dump sequence numbers if there have been non-logged writes. This prevents you from using an earlier database dump or external copy as an improper foundation for a dump sequence.

For example, in the warm standby method that is described in Figure 27-5, archives are produced by dump database (D1), dump transaction (T1), quiesce database, dump transaction (T2), and dump transaction (T3):

Figure 27-5: Warm standby dump sequence



Typically, in an environment with logged updates and no dump tran with truncate_only, you could load D1, T1, T2, and T3 in turn, bypassing any quiesce database hold. This approach is used in a warm standby situation, where succeeding database dumps on the primary server simplify media failure recovery scenarios. On the secondary, or standby server, which is used for decision support systems, you may prefer continuous incremental applications of load transaction instead of interruptions from external copy operation.

However, if an unlogged operation occurs (say, a select into, as happens in Figure 27-5) after the dump transaction that produces T1, a subsequent dump transaction to archive is not allowed, and you must either create another dump of the database, or issue quiesce database for external copy and then make a new external copy of the database. Issuing either of these commands updates the dump sequence number and clears the mark that blocks the dump transaction to archive.

Whether or not you use the for external dump clause depends on how you want recovery to treat the quiescent database that would be marked as `in quiesce`.

quiesce database hold

If you issue quiesce database and do not use the for external dump clause, during the external copy operation that creates the secondary set of databases, the secondary server is not running, and recovery under -q will not see any copied database as “in quiesce.” It will recover each server in the normal fashion during start-up recovery; it will not recover them as for load database as was previously described. Subsequently, any attempt to perform a load tran to any of these databases is disallowed with error 4306, `"There was activity on database since last load ..."`, or with error 4305, `"Specified file '%.*s' is out of sequence ..."`

Whether or not there been unlogged activity in the primary database, the dump sequence number will not incremented by quiesce database hold, and the unlogged-writes bits are not cleared by quiesce database release.

quiesce database hold for external dump

When you issue quiesce database for external dump, the external copy of the database “remembers” that it was made during a quiescent interval, so that -q recovery can recover it, as happens for load database. quiesce database release clears this information from the primary database. If non-logged writes have prevented dump tran *to archive* on the primary server, dump tran *to archive* is now enabled.

For any database in quiesce database's list, if non-logged writes have occurred since the previous dump database or quiesce database hold for external dump, the dump sequence number is updated by quiesce database hold for external dump, and the non-logged-writes information is cleared by quiesce database release. The updated sequence number causes load tran to fail if it is applied to a target other than the external copy created under the quiesce database that updated it. This resembles the behavior for dump database of a database with non-logged writes status.

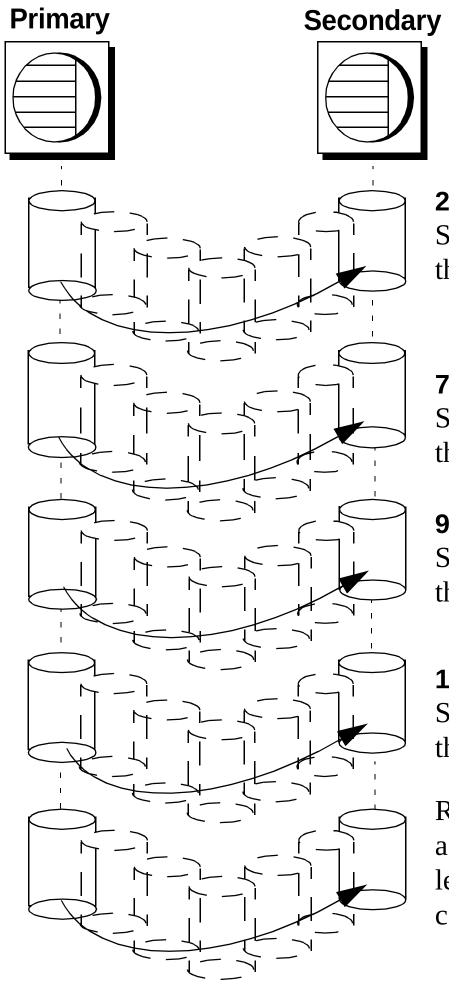
Warning! quiesce database for external dump clears the internal flag that prevents you from performing dump transaction to *archive_device* whether or not you actually make an external copy or perform a database dump. quiesce database has no way of knowing whether or not you have made an external copy. *It is incumbent upon you to perform this duty*. If you use quiesce database hold for external dump to effect a transient write protection rather than to actually perform a copy that serves as the foundation for a new dump sequence, and your application includes occasional unlogged writes, Adaptive Server may allow you to create transaction log dumps that cannot be used. In this situation, dump transaction to *archive_device* initially succeeds, but future load transaction commands may reject these archives because they are out of sequence.

Backing up primary devices with *quiesce database*

Typically, users back up their databases with quiesce database using one of the following methods. Both allow you to off-load decision support applications from the online transaction processor (OLTP) server during normal operation:

- Iterative refresh of the primary device – copy the primary device to the secondary device at refresh intervals. Quiesce the database before each refresh. A system that provides weekly backups using this system is shown in Figure 27-6:

Figure 27-6: Backup schedule for iterative refresh method



2:00 a.m.
1) Issue “quiesce database hold.”
2) Copy databases using external command.
3) Issue “quiesce database release.”

2:10 a.m.
Start the server without the -q option.”

7:00 a.m.
Perform the steps above.

7:10 a.m.
Start the server without the -q option..

9:00 a.m.
Perform the steps above.

9:10 a.m.
Start the server without the -q option.

10:00 a.m.
Perform the steps above.

10:10 a.m.
Start the server without the -q option.

Repeat each hour until activity tapers off, then lengthen intervals accordingly.

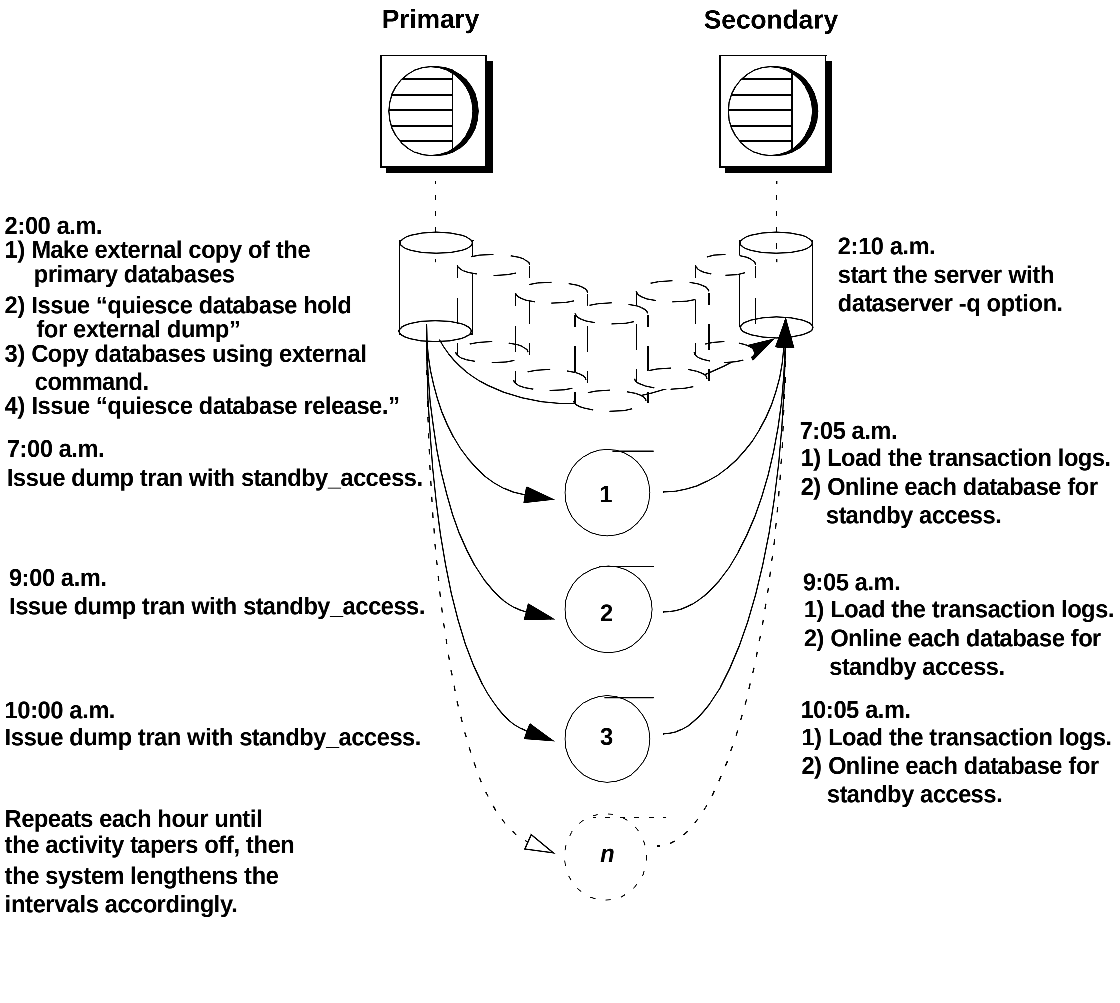
Repeat each hour until activity tapers off, then lengthen intervals accordingly.

If you are using the iterative refresh method, you do not have to use the -q option to restart the secondary server (after a crash or system maintenance). Any incomplete transactions generate compensation log records, and the affected databases come online in the regular fashion.

- Warm standby method – allows full concurrency for the OLTP server because it does not block writes.

After you make an external (secondary) copy of the primary database devices using the for external dump clause, refresh the secondary databases with periodic applications of the transaction logs with dumps from the primary server. For this method, quiesce the databases once to make an external copy of the set of databases and then refresh each periodically using a dump tran with standby_access. A system that uses a daily update of the primary device and then hourly backups of the transaction log is shown in Figure 27-7.

Figure 27-7: Backup schedule for warm standby method


Primary
Secondary
2:00 a.m.
1) Make external copy of the primary databases
2) Issue “quiesce database hold for external dump”
3) Copy databases using external command.
4) Issue “quiesce database release.”
2:10 a.m.
start the server with dataserver -q option.
7:00 a.m.
Issue dump tran with standby_access.
1
7:05 a.m.
1) Load the transaction logs.
2) Online each database for standby access.
9:00 a.m.
Issue dump tran with standby_access.
2
9:05 a.m.
1) Load the transaction logs.
2) Online each database for standby access.
10:00 a.m.
Issue dump tran with standby_access.
3
10:05 a.m.
1) Load the transaction logs.
2) Online each database for standby access.
Repeats each hour until the activity tapers off, then the system lengthens the intervals accordingly.
n

Recovery of databases for warm standby method

If you are using the warm standby method, Adaptive Server must know whether it is starting the primary or the secondary server. Use the -q option of the dataserver command to specify that you are starting the secondary server. If you do not start the server with the -q option:

- The databases are recovered normally, not as they would be for load database.
- Any uncommitted transactions at the time you issue quiesce database are rolled back.

See “Starting the secondary server with the -q option” on page 847 for more information.

The recovery sequence proceeds differently, depending on whether the database is marked `in quiesce`.

Recovery of databases that are not marked “in quiesce”

Under the -q option, if a database is not marked `in quiesce`, it is recovered as it would be in the primary server. That is, if the database is not currently in a load sequence from previous operations, it is fully recovered and brought online. If there are incomplete transactions, they are rolled back and compensation log records are written during recovery.

Recovery of databases that are marked as “in quiesce”

- User databases – user databases that are marked `in quiesce` recover in the same manner as databases recovering during load database. This enables load tran to detect any activity that has occurred in the primary database since the server was brought down. After you start the secondary server with the -q option, the recovery process encounters the `in quiesce` mark. Adaptive Server issues a message stating that the database is in a load sequence and is being left offline. If you are using the warm standby method, do not bring the database online for its decision support system role until you have loaded the first transaction dump produced by a dump tran with standby_access. Then use online database for standby_access.
- System databases – system databases come fully online immediately. The `in quiesce` mark is erased and ignored.

Making archived copies during the quiescent state

quiesce database hold for external dump signifies your intent to make external copies of your databases during the quiescent state. Because these external copies are made after you issue quiesce database hold, the database is transactionally consistent because you are assured that no writes occurred during the interval between the quiesce database hold and the quiesce database release, and recovery can be run in the same manner as start-up recovery. This process is described in Figure 27-5.

If the environment does not have unlogged updates and does not include a dump tran with truncate_only, you might load D1, T1, T2, and T3 in turn, bypassing any quiesce database...hold commands. However, if an unlogged operation (such as a select into shown in Figure 27-5) occurs after the dump transaction that produces T1, dump transaction to archive is no longer allowed.

Using the quiesce database hold for external dump clause addresses this problem by clearing the status bits that prevent the next dump transaction to archive and changing the sequence number of the external copy to create a foundation for a load sequence. However, if there have been no non-logged writes, the sequence number is not incremented.

With or without the for external dump clause, you can make external copies of the databases. However, to apply subsequent transaction dumps from the primary to the secondary servers, you must include the for external dump clause, as shown:

```
quiesce database tag_name hold db_name [, db_name]
... [for external dump]
```

For example:

```
quiesce database pubs_tag hold pubs2 for external
dump
```

Assuming the database mappings have not changed since the primary instance of the database were initiated, the steps for making an external dump are for a single database include:

1 Issue the quiesce database hold for external dump command:

   ```
   quiesce database pubs_tag hold pubs2 for external
   dump
   ```

2 Make an external copy of the database using the method appropriate for your site.

3 Issue quiesce database *tag_name* release:

```
quiesce database pubs_tag release
```

Warning! Clearing the status bits and updating the sequence number enables you to perform a dump transaction whether or not you actually make an external copy after you issue quiesce database. Adaptive Server has no way of knowing whether or not you have made an external copy during the time between quiesce database... hold for external dump and quiesce database... release. If you use the quiesce database hold for external dump command to effect a transient write protection rather than to actually perform a copy that can serve as the foundation for a new dump sequence, and your application includes occasional unlogged writes, Adaptive Server allows you to create transaction log dumps that cannot be used. dump transaction to *archive_device* succeeds, but load transaction rejects these archives as being out of sequence.

Using *mount* and *unmount* commands

The mount and unmount commands make it easier to move or copy databases. You can move or copy a database from one Adaptive Server to another without rebooting the server. You can move or copy more than one database at a time using the mount and unmount commands.

These commands can also be used when you need to physically move the devices and then reactivate the databases.

Warning! Direct mapping to a login name is not maintained within a database in Adaptive Server. This means that, for every login allowed access to a database on the original Adaptive Server, a corresponding login for the same suid must exist at the destination Adaptive Server.

For permissions and protections to remain unchanged the login maps at the secondary Adaptive Server must be identical to the files on the first Adaptive Server.

Manifest file

The *manifest* file is the binary file which describes the databases that are present on a set of database devices. It can be created only if the set of databases that occupy those devices are isolated, self-contained on those devices.

Since the *manifest* is a binary file, operations that can do character translations of the file contents (such as ftp) will corrupt the file unless done in binary mode

Performance considerations

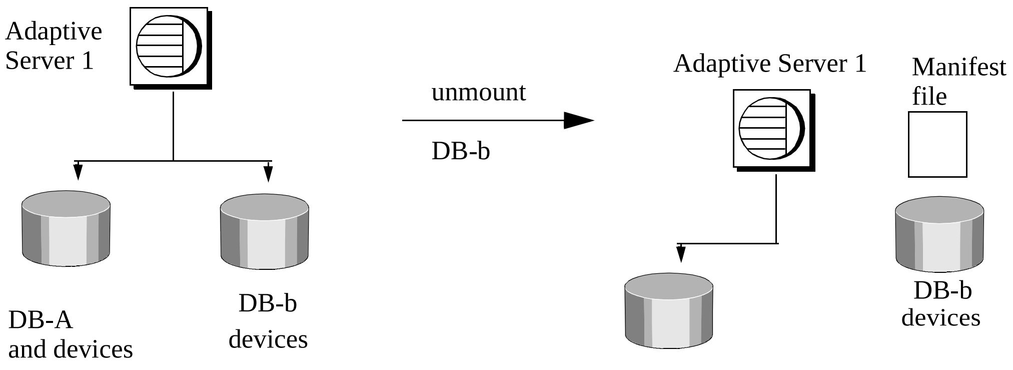
A change in dbid results in all procedures being marked for recompiling in the database. This affects the time taken to recover the database at the destination and the first execution of the procedure.

System restrictions

- System databases cannot be unmounted. However, unmounting of sybsystemprocs is allowed.
- Proxy databases cannot be unmounted.
- mount and unmount database commands are not allowed in a transaction.
- mount database is not allowed in an HA configured server.

Unmounting the database

Figure 27-8: unmount database command



Warning! The unmount command removes a database and all its information from the Adaptive Server. Use the unmount command with extreme caution.

When you unmount a database, you remove the database and its devices from an Adaptive Server. The unmount command shuts down the database and drops it from the Adaptive Server. The devices are also deactivated and dropped. The database and its pages are not altered when they are unmounted. The database pages will still be present on the OS devices.

```
unmount database <dbname list> to <manifest_file>
```

Use quiesce database to create the *manifest* file, then execute unmount. For example:

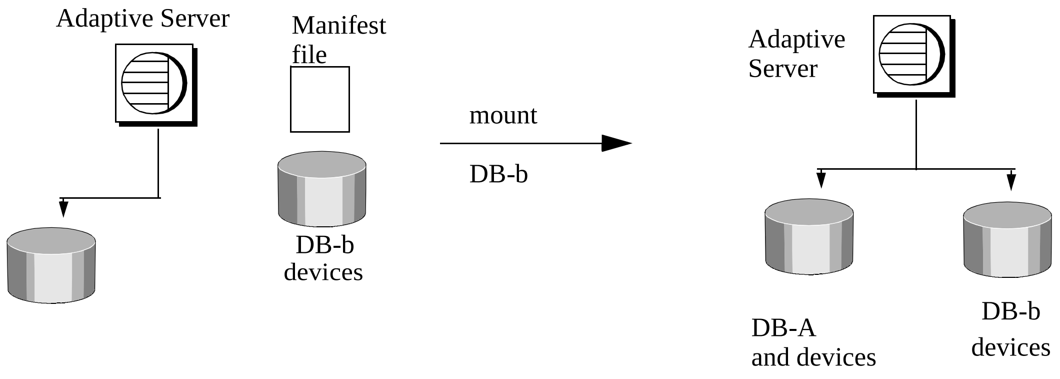
```
1> unmount database pubs2 to
"/work2/Devices/Mpubs2_file"
2> go
```

If you now try to use the pubs2 database you see this error:

```
Attempt to locate entry in sysdatabases for database
'pubs2' by name failed - no entry found under that
name. Make sure that name is entered properly.
```

Mounting a database

Figure 27-9: mount command



You use the mount command to add the information for devices and other attributes for the database to the destination or secondary Adaptive Server. The mount command decodes the information in the *manifest* file and makes the set of databases available online. All the required supporting activities are done, including adding database devices if not present and activating them, creating the catalog entries for the new databases, recovering them and putting them online. When you mount databases onto an Adaptive Server:

- It is not possible to mount a subset of the databases described in the manifest. All the databases and devices present in the manifest have to be mounted together.
- The databases being mounted must have the same page size as the previous Adaptive Server.
- There must be enough devices configured on the secondary Adaptive Server for the successful addition of all the devices belonging to the mounted databases.
- The configuration parameter number of devices must be set appropriately
- Database names and devices with the same names as the mounted database must not already exist.
- Adaptive Server must have the same version as the mounted database.

- The mounted database must be from the same platform as the Adaptive Server.

Note Remember that if you use the unmount command, the database is removed from the original Adaptive Server with its information on attributes, device names, and so on.

Syntax:

```
mount all from <manifest_file>
```

For example:

```
1> mount database all from
"/work2/Devices/Mpubs_file"
2> go

Redo pass of recovery has processed 1 committed and
0 aborted transactions.
MOUNT DATABASE: Completed recovery of mounted
database 'pubs2'
```

Once the mount is completed, the database is still offline. Use the online database command to bring the database online. You do not have to reboot the server.

Renaming devices

The *manifest* file contains the device paths as known to the Adaptive Server which created the *manifest* file. If the mounting Adaptive Server accesses the devices with a different path, you can specify the new path to the mount command.

1 Use the mount command with the listonly to get the old path:

```
1> mount database all from "/work2/Mpubs_file"
with listonly
2> go

"/work2/Devices/pubsdat.dat" = "pubs2dat"
```

2 If the new path for the device *pubs2dat* is /work2/Devices/pubsdevice.dat specify the new device in the mount command:

```
mount database all from "/work2/Mpubs_file" using
"work2/datadevices/pubsdevice.dat" = "pubs2dat"
```

quiesce database

You use the quiesce database hold command with the *manifest* clause to hold the database and create a *manifest* file.

Note Since the *manifest* file is binary, operations that perform character translations of the file contents (such as ftp) corrupt the file unless performed in binary mode.

```
quiesce database < tag_name > hold < dbname list > [for external dump] [ to
<manifest_file> [with override]]
```

For example:

Place the database in a hold status and build the *manifest* file:

```
quiesce database pubs2_tag hold pubs2 for external
dump to "/work2/Devices/Mpubs_file"
```

This stops transactions against the database and creates the *manifest* file. To verify the devices within the manifest file, you can execute the mount with listonly:

```
mount database all from “/work2/Devices/Mpubs_file”
with listonly

“/work2/Devices/pubsdat.dat” = “pubs2dat”
```

You execute the quiesce database hold and then copy the database devices.

A *manifest* file cannot be created if the set of databases that are quiesced contain references to databases outside of the set. You may use the override option to bypass this restriction.

Creating a mountable copy of a database

1 Use the quiesce database command with the manifest clause and quiesce the database. This command will create a manifest file describing the database.

2 Use the mount command with listonly to get the list of devices to be copied.

3 Use external copy utilities, such as cp, dd, split mirror, and so on, to copy the database devices to another Adaptive Server.

The copy of the devices and the *manifest* file is a mountable copy of the database.

Moving databases from one Adaptive Server to another

1. Use the unmount command to unmount the database from the first Adaptive Server. The command will create a manifest file describing the database.
2. Make the database devices available to the second Adaptive Server, if not already available. This may require the help of your OS administrator if the second Adaptive Server is on another machine.
3. Execute the mount command on the secondary Adaptive Server with the manifest file created in Step 1.

Designating responsibility for backups

Many organizations have an operator who performs all backup and recovery operations. Only a System Administrator, a Database Owner, or an operator can execute the dump and load commands. The Database Owner can dump only his or her own database. The operator and System Administrator can dump and load any database.

Any user can execute sp_volchanged to notify the Backup Server when a tape volume is changed.

Using the Backup Server for backup and recovery

Dumps and loads are performed by an Open Server program, Backup Server, running on the same machine as Adaptive Server. You can perform backups over the network, using a Backup Server on a remote computer and another on the local computer.

Note Backup Server cannot dump to multi-disk volumes.

Backup Server:

- Creates and loads from “striped dumps.” **Dump striping** allows you to use up to 32 backup devices in parallel. This splits the database into approximately equal portions and backs up each portion to a separate device.
- Creates and loads single dumps that span several tapes.
- Dumps and loads over the network to a Backup Server running on another machine.
- Dumps several databases or transaction logs onto a single tape.
- Loads a single file from a tape that contains many database or log dumps.
- Supports platform-specific tape handling options.
- Directs volume-handling requests to the session where the dump or load command was issued or to its operator console.
- Detects the physical characteristics of the dump devices to determine protocols, block sizes, and other characteristics.

Relationship between Adaptive Server and Backup Servers

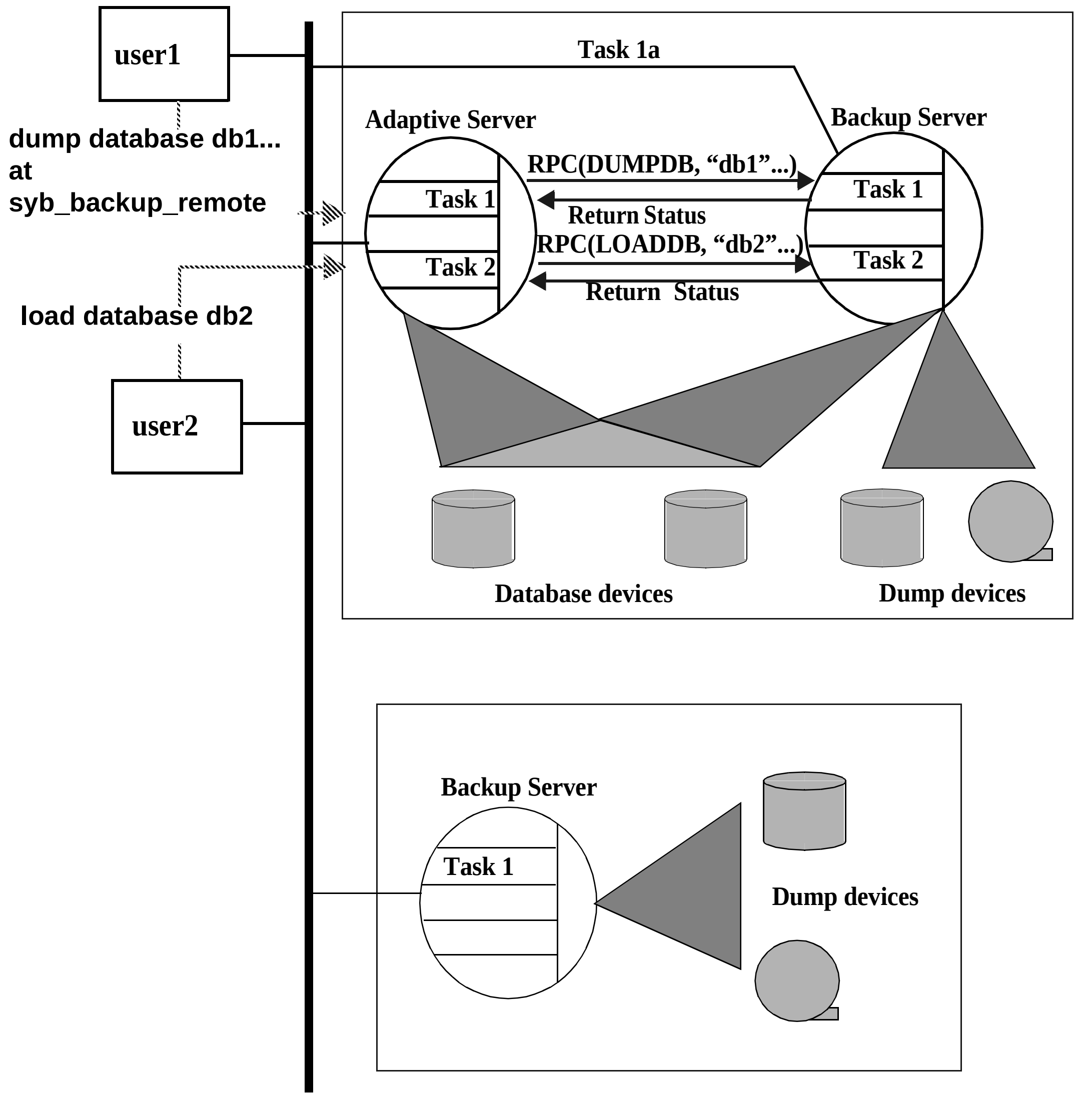
Figure 27-10 shows two users performing backup activities simultaneously on two databases:

- User1 is dumping database `db1` to a remote Backup Server.
- User2 is loading database `db2` from the local Backup Server.

Each user issues the appropriate dump or load command from a Adaptive Server session. Adaptive Server interprets the command and sends remote procedure calls (RPCs) to the Backup Server. The calls indicate which database pages to dump or load, which dump devices to use, and other options.

While the dumps and loads execute, Adaptive Server and Backup Server use RPCs to exchange instructions and status messages. Backup Server—not Adaptive Server—performs all data transfer for the dump and load commands.

Figure 27-10: Adaptive Server and Backup Server with remote Backup Server



When the local Backup Server receives user1's dump instructions, it reads the specified pages from the database devices and sends them to the remote Backup Server. The remote Backup Server saves the data to offline media.

Simultaneously, the local Backup Server performs user2's load command by reading data from local dump devices and writing it to the database device.

Communicating with the Backup Server

To use the dump and load commands, an Adaptive Server must be able to communicate with its Backup Server. These are the requirements:

- The Backup Server must be running on the same machine as the Adaptive Server (or on the same cluster for OpenVMS).
- The Backup Server must be listed in the master..sysservers table. The Backup Server entry, SYB_BACKUP, is created in sysservers when you install Adaptive Server. Use sp_helpserver to see this information.
- The Backup Server must be listed in the interfaces file. The entry for the local Backup Server is created when you install Adaptive Server. The name of the Backup Server listed in the interfaces file must match the column srvnet name for the SYB_BACKUP entry in master..sysservers. If you have installed a remote Backup Server on another machine, create the interfaces file on a file system that is shared by both machines, or copy the entry to your local interfaces file. The name of the remote Backup Server must be the same in both interfaces files.
- The user who starts the Backup Server process must have write permission for the dump devices. The "sybase" user, who usually starts Adaptive Server and Backup Server, can read from and write to the database devices.
- Adaptive Server must be configured for remote access. By default, Adaptive Server is installed with remote access enabled. See "Configuring your server for remote access" on page 867 for more information.

Mounting a new volume

Note Backup Server cannot dump to multi-disk volumes.

During the backup and restore process, change tape volumes. If the Backup Server detects a problem with the currently mounted volume, it requests a volume change by sending messages to either the client or its operator console. After mounting another volume, the operator notifies the Backup Server by executing sp_volchanged on Adaptive Server.

On UNIX systems, the Backup Server requests a volume change when the tape capacity has been reached. The operator mounts another tape and then executes sp_volchanged (see Table 27-2).

Table 27-2: Changing tape volumes on a UNIX system

Sequence	Operator using isql	Adaptive Server	Backup Server
1	Issues the dump database command		
2		Sends dump request to Backup Server	
3			Receives dump request message from Adaptive Server Sends message for tape mounting to operator Waits for operator's reply
4	Receives volume change request from Backup Server Mounts tapes Executes sp_volchanged		
5			Checks tapes If tapes are okay, begins dump When tape is full, sends volume change request to operator
6	Receives volume change request from Backup Server Mounts tapes Executes sp_volchanged		
7			Continues dump When dump is complete, sends messages to operator and Adaptive Server

Sequence	Operator using isql	Adaptive Server	Backup Server
8	Receives message that dump is complete Removes and labels tapes	Receives message that dump is complete Releases locks Completes the dump database command	

Starting and stopping Backup Server

Most UNIX systems use the startserver utility to start Backup Server on the same machine as Adaptive Server. On Windows NT, you can start Backup Server from Sybase Central. See the configuration documentation for your platform for information about starting Backup Server.

Use shutdown to shut down a Backup Server. See Chapter 5, “Diagnosing System Problems,” and the *Reference Manual* for information about this command.

Configuring your server for remote access

The remote access configuration parameter is set to 1 when you install Adaptive Server. This allows Adaptive Server to execute remote procedure calls to the Backup Server.

For security reasons, you may want to disable remote access except when dumps and loads are taking place. To disable remote access, use:

```
sp_configure "allow remote access", 0
```

Before you perform a dump or load, use the following command to re-enable remote access:

```
sp_configure "allow remote access", 1
```

allow remote access is dynamic and does not require a restart of Adaptive Server to take effect. Only a System Security Officer can set allow remote access.

Choosing backup media

Tapes are preferred as dump devices, since they permit a library of database and transaction log dumps to be kept offline. Large databases can span multiple tape volumes. On UNIX systems, the Backup Server requires nonrewinding tape devices for all dumps and loads.

For a list of supported dump devices, see the the configuration documentation for your platform.

Protecting backup tapes from being overwritten

The tape retention in days configuration parameter determines how many days' backup tapes are protected from being overwritten. The default value of tape retention in days is 0. Which means that backup tapes can be overwritten immediately.

Use sp_configure to change the tape retention in days value. The new value takes effect the next time you restart Adaptive Server:

```
sp_configure "tape retention in days", 14
```

Both dump database and dump transaction provide a retaindays option that overrides the tape retention in days value for that dump.

Dumping to files or disks

In general, dumping to a file or disk is not recommended. If the disk or computer containing that file crashes, there may be no way to recover the dumps. On UNIX and PC systems, the entire master database dump must fit into a single volume. On these systems, dumping to a file or disk is your only option if the master database is too large to fit on a single tape volume, unless you have a second Adaptive Server that can issue sp_volchanged requests.

Dumps to a file or disk can be copied to tape for offline storage, but these tapes must be copied back to an online file before they can be read by Adaptive Server. Backup Server cannot directly read a dump that is made to a disk file and then copied to tape.

Creating logical device names for local dump devices

If you are dumping to or loading from local devices (that is, if you are not performing backups over a network to a remote Backup Server), you can specify dump devices either by providing their physical locations or by specifying their logical device names. In the latter case, you may want to create logical dump device names in the sysdevices system table of the master database.

Note If you are dumping to or loading from a remote Backup Server, you must specify the absolute path name of the dump device. You cannot use a logical device name.

The sysdevices table stores information about each database and backup device, including its *physical_name* (the actual operating system device or file name) and its *device_name* (or logical name, known only within Adaptive Server). On most platforms, Adaptive Server has one or two aliases for tape devices installed in sysdevices. The physical names for these devices are common disk drive names for the platform; the logical names are tapedump1 and tapedump2.

When you create backup scripts and threshold procedures, use logical names, rather than physical device names, and whenever possible, you must modify scripts and procedures that refer to actual device names each time you replace a backup device. If you use logical device names, you can simply drop the sysdevices entry for the failed device and create a new entry that associates the logical name with a different physical device.

Note Make sure that the device driver options you include with the dump command are accurate. Backup Server does not verify any device driver options you include during a dump command. For example, if you include a option that forces Backup Server to rewind a tape before use, it will always rewind the tape to the beginning instead of reading the tape from the point of the dump

---

Listing the current device names

To list the backup devices for your system, run:

```
select * from master..sysdevices
```

```
   where status = 16 or status = 24
```

To list both the physical and logical names for database and backup devices, use sp_helpdevice:

```
sp_helpdevice tapedump1
device_name physical_name
 description
 status cntrltype device_number low      high
 ------ --------- ------------- -------- -------
tapedump1 /dev/nrmt4
tape, 625 MB, dump device
   16         3             0        0   20000
```

Adding a backup device

Use sp_addumpdevice to add a backup device:

sp_addumpdevice{ "tape" | "disk"} , *logicalname*, *physicalname*, *tapesize*

where:

physicalname can be either an absolute path name or a relative path name. During dumps and loads, the Backup Server resolves relative path names by looking in Adaptive Server's current working directory.

tapesize is the capacity of the tape in megabytes. Most platforms require this parameter for tape devices but ignore it for disk devices. The Backup Server uses the *tapesize* parameter if the dump command does not specify a tape capacity.

tapesize must be at least 1MB and should be slightly below the capacity rated for the device.

Redefining a logical device name

To use an existing logical device name for a different physical device, drop the device with sp_dropdevice and then add it with sp_addumpdevice. For example:

```
sp_dropdevice tapedump2
sp_addumpdevice "tape", tapedump2, "/dev/nrmt8", 625
```

Scheduling backups of user databases

A major task in developing a backup plan is determining how often to back up your databases. The frequency of your backups determines how much work you will lose in the event of a media failure. This section presents some guidelines about when to dump user databases and transaction logs.

Scheduling routine backups

Dump each user database just after you create it, to provide a base point, and on a fixed schedule thereafter. Daily backups of the transaction log and weekly backups of the database are the minimum recommended. Many installations with large and active databases make database dumps every day and transaction log dumps every half hour or hour.

Interdependent databases—databases where there are cross-database transactions, triggers, or referential integrity—should be backed up at the same time, during a period when there is no cross-database data modification activity. If one of these databases fails and needs to be reloaded, they should all be reloaded from these simultaneous dumps.

Warning! Always dump both databases immediately after adding, changing, or removing a cross-database constraint or dropping a table that contains a cross-database constraint.

Other times to back up a database

In addition to routine dumps, you should dump a database each time you upgrade a user database, create a new index, perform an unlogged operation, or run the dump transaction with no_log or dump transaction with truncate_only command.

Dumping a user database after upgrading

After you upgrade a user database to the current version of Adaptive Server, dump the newly upgraded database to create a dump that is compatible with the current release. A dump database must occur on upgraded user databases before a dump transaction is permitted.

Dumping a database after creating an index

When you add an index to a table, create index is recorded in the transaction log. As it fills the index pages with information, however, Adaptive Server does not log the changes.

If your database device fails after you create an index, load transaction may take as long to reconstruct the index as create index took to build it. To avoid lengthy delays, dump each database immediately after creating an index on one of its tables.

Dumping a database after unlogged operations

Adaptive Server writes the data for the following commands directly to disk, adding no entries (or, in the case of bcp, minimal entries) in the transaction log:

- Non-logged writetext
- select into on a permanent table
- Fast bulk copy (bcp) into a table with no triggers or indexes

You cannot recover any changes made to the database after issuing one of these commands. To ensure that these commands are recoverable, issue a dump database command immediately after executing any of these commands.

Dumping a database when the log has been truncated

dump transaction with truncate_only and dump transaction with no_log remove transactions from the log without making a backup copy. To ensure recoverability, you must dump the database each time you run either command because of lack of disk space. You cannot copying the transaction log until you have done so. See "Using the special dump transaction options" on page 841 for more information.

If the trunc log on chkpt database option is set to true, and the transaction log contains 50 rows or more, Adaptive Server truncates the log when an automatic checkpoint occurs. If this happens, you must dump the entire database—not the transaction log—to ensure recoverability.

Scheduling backups of *master*

Back up the master database *regularly and frequently.*

Backups of the master database are used as part of the recovery procedure in case of a failure that affects the master database. If you do not have a current backup of master, you may have to reconstruct vital system tables at a time when you are under pressure to get your databases up and running again.

Dumping *master* after each change

Although you can restrict the creation of database objects in master, system procedures such as sp_addlogin and sp_droplogin, sp_password, and sp_modifylogin allow users to modify system tables in the database. Back up the master database frequently to record these changes.

Back up the master database after each command that affects disks, storage, databases, or segments. Always back up master after issuing any of the following commands or system procedures:

- disk init, sp_addumpdevice, or sp_dropdevice
- Disk mirroring commands
- The segment system procedures sp_addsegment, sp_dropsegment, or sp_extendsegment
- create procedure or drop procedure
- sp_logdevice
- sp_configure
- create database or alter database

Saving scripts and system tables

For further protection, save the scripts containing all of your disk init, create database, and alter database commands and make a hard copy of your sysdatabases, sysusages, and sysdevices tables each time you issue one of these commands.

You cannot use the dataserver command to automatically recover changes that result from these commands. If you keep your scripts—files containing Transact-SQL statements—you can run them to re-create the changes. Otherwise, you must reissue each command against the rebuilt master database.

You should also keep a hard copy of syslogins. When you recover master from a dump, compare the hard copy to your current version of the table to be sure that users retain the same user IDs.

For information on the exact queries to run against the system tables, see “Backing up master and keeping copies of system tables” on page 24.

Truncating the *master* database transaction log

Since the master database transaction log is on the same database devices as the data, you cannot back up its transaction log separately. You cannot move the log of the master database. You must always use dump database to back up the master database. Use dump transaction with the truncate_only option periodically (for instance, after each database dump) to purge the transaction log of the master database.

Avoiding volume changes and recovery

When you dump the master database, be sure that the entire dump fits on a single volume, unless you have more than one Adaptive Server that can communicate with your Backup Server. You must start Adaptive Server in single-user mode before loading the master database. This does not allow a separate user connection to respond to Backup Server’s volume change messages during the load. Since master is usually small in size, placing its backup on a single tape volume is typically not a problem.

Scheduling backups of the *model* database

Keep a current database dump of the model database. Each time you make a change to the model database, make a new backup. If model is damaged and you do not have a backup, you must reenter all the changes you have made to restore model.

Truncating the *model* database's transaction log

model, like master, stores its transaction log on the same database devices as the data. You must always use dump database to back up the model database and dump transaction with truncate_only to purge the transaction log after each database dump.

Scheduling backups of the *sybsystemprocs* database

The sybsystemprocs database stores only system procedures. Restore this database by running the installmaster script, unless you make changes to the database.

If you change permissions on some system procedures, or create your own system procedures in sybsystemprocs, your two recovery choices are:

- Run installmaster, then reenter all of your changes by re-creating your procedures or by re-executing the grant and revoke commands.
- Back up sybsystemprocs each time you make a change to it.

Both of these recovery options are described in Chapter 29, "Restoring the System Databases."

Like other system databases, sybsystemprocs stores its transaction log on the same device as the data. You must always use dump database to back up sybsystemprocs. By default, the trunc log on chkpt option is set to true (on) in sybsystemprocs, so you should not need to truncate the transaction log. If you change this database option, be sure to truncate the log when you dump the database.

If you are running on a UNIX system or PC, and you have only one Adaptive Server that can communicate with your Backup Server, be sure that the entire dump of sybsystemprocs fits on a single dump device. Signaling volume changes requires sp_volchanged, and you cannot use this procedure on a server while sybsystemprocs is in the process of recovery.

Configuring Adaptive Server for simultaneous loads

Adaptive Server can perform multiple load and dump commands simultaneously. Loading a database requires one 16K buffer for each active database load. By default, Adaptive Server is configured for six simultaneous loads. To perform more loads simultaneously, a System Administrator can increase the value of number of large i/o buffers:

```
sp_configure "number of large i/o buffers", 12
```

This parameter requires a restart of Adaptive Server. See "number of large i/o buffers" on page 71 for more information. These buffers are not used for dump commands or for load transaction.

Gathering backup statistics

Use dump database to make several practice backups of an actual user database and dump transaction to back up a transaction log. Recover the database with load database and apply successive transaction log dumps with load transaction.

Keep statistics on how long each dump and load takes and how much space it requires. The more closely you approximate real-life backup conditions, the more meaningful your predictions will be.

After you have developed and tested your backup procedures, commit them to paper. Determine a reasonable backup schedule and adhere to it. If you develop, document, and test your backup procedures ahead of time, you will be much better prepared to get your databases online if disaster strikes.

CHAPTER 28

Backing Up and Restoring User Databases

Regular and frequent backups are your only protection against database damage that results from failure of your database devices.

Dump and load command syntax

The dump database, dump transaction, load database, and load transaction commands have parallel syntax. Routine dumps and loads require the name of a database and at least one dump device. The commands can also include the following options:

- compress::*compression_level*:: to compress your dump files
- at *server_name* to specify the remote Backup Server
- density, blocksize, and capacity to specify tape storage characteristics
- dumpvolume to specify the volume name of the ANSI tape label
- file = *file_name* to specify the name of the file to dump to or load from
- stripe on *stripe_device* to specify additional dump devices
- dismount, unload, init, and retaindays to specify tape handling
- notify to specify whether Backup Server messages are sent to the client that initiated the dump or load or to the operator_console

Note When a user database is dumped, its database options are not dumped because they are stored in the sysdatabases table of the master databases. This is not a problem if you load a previously dumped database onto itself because rows in sysdatabases describing this database still exist in master. However, if you drop the database before you perform the load database, or if you load the database dump on a new server, these database options are not restored. To restore the image of a user database, you must also recreate the database options.

Table 28-1 shows the syntax for routine database and log dumps and for dumping the log after a device failure. It indicates what type of information is provided by each part of the dump database or dump transaction statement.

Table 28-1: Syntax for routine dumps and log dumps after device failure

Information provided	**Task**	
	Routine database or log dump	**Log dump after device failure**
Command	`dump {database | transaction}`	`dump transaction`
Database name	*`database_name`*	*`database_name`*

Information provided	**Task**	
	Routine database or log dump	**Log dump after device failure**
Compression	`to [compress::[`*`compression_level`*`::]]`	`to [compress::[`*`compression_level`*`::]]`
Dump device	*`stripe_device`*	*`stripe_device`*
Remote Backup Server	`[at `*`server_name`*`]`	`[at `*`server_name`*`]`
Tape device characteristics	`[density = `*`density`*`, blocksize = `*`number_bytes`*`, capacity = `*`number_kilobytes`*`]`	`[density = `*`density`*`, blocksize = `*`number_bytes`*`, capacity = `*`number_kilobytes`*`]`
Volume name	`[, dumpvolume = `*`volume_name`*`]`	`[, dumpvolume = `*`volume_name`*`]`
File name	`[, file = `*`file_name`*`]`	`[, file = `*`file_name`*`]`
Characteristics of additional devices (up to 31 devices; one set per device)	`[stripe on [compress::[`*`compression_level`*`::]] stripe_device [at `*`server_name`*`] [density = `*`density`*`, blocksize = `*`number_bytes`*`, capacity = `*`number_kilobytes`*`, file = `*`file_name`*`, dumpvolume = `*`volume_name`*`]]...`	`[stripe on [compress::[`*`compression_level`*`::]] stripe_device [at `*`server_name`*`] [density = `*`density`*`, capacity = `*`number_kilobytes`*`, file = `*`file_name`*`, dumpvolume = `*`volume_name`*`]]...`
Options that apply to entire dump	`[with { density = `*`density`*`, blocksize = `*`number_bytes`*`, capacity = `*`number_kilobytes`*`, file = `*`file_name`*`, [nodismount | dismount], [nounload | unload], [retaindays = `*`number_days`*`], [noinit | init], file = `*`file_name`*`, dumpvolume = `*`volume_name`*` standby_access`	`[with { density = `*`density`*`, blocksize = `*`number_bytes`*`, capacity = `*`number_kilobytes`*`, file = `*`file_name`*`, [nodismount | dismount], [nounload | unload], [retaindays = `*`number_days`*`], [noinit | init], file = `*`file_name`*`, dumpvolume = `*`volume_name`*`, standby_access`
Do not truncate log		`no_truncate`
Message destination	`[, notify = {client | operator_console}]}]`	`[, notify = {client | operator_console}]}]`

Table 28-2 shows the syntax for loading a database, applying transactions from the log, and returning information about dump headers and files.

Table 28-2: Syntax for load commands

Information provided	**Task**	
	Load database or apply recent transactions	**Return header or file information but do not load backup**
Command	`load {database | transaction}`	`load {database | transaction}`
Database name	*`database_name`*	*`database_name`*
Compression	`from [compress::]`	`from [compress::]`
Dump device	*`stripe_device`*	*`stripe_device`*
Remote Backup Server	`[at` *`server_name`*`]`	`[at` *`server_name`*`]`
Tape device characteristics	`[density =` *`density`*`,`	`[density =` *`density`*`,`
Volume name	`[, dumpvolume =` *`volume_name`*`]`	`[, dumpvolume =` *`volume_name`*`]`
File name	`[, file =` *`file_name`*`]`	`[, file =` *`file_name`*`]`
Characteristics of additional devices (up to 31 devices; one set per device)	`[stripe on [compress::]`*`stripe_device`* `[at` *`server_name`*`] [density =` *`density`*`, file =` *`file_name`*`, dumpvolume =` *`volume_name`*`]]...`	`[stripe on [compress::]`*`stripe_device`* `[at` *`server_name`*`] [density =` *`density`*`, file =` *`file_name`*`, dumpvolume =` *`volume_name`*`]]...`
Tape handling	`[with{ [density =` *`density`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`, [nodismount | dismount], [nounload | unload]`	`[with{ [density =` *`density`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`, [nodismount | dismount], [nounload | unload]`
Provide header information		`[, headeronly]`
List dump files		`[, listonly [= full]]`
Message destination	`[, notify = {client | operator_console}]}]`	`[, notify = {client | operator_console}]}]`
Do not load open transactions		

Table 28-3 shows the syntax for truncating a log:

- That is not on a separate segment
- Without making a backup copy
- With insufficient free space to successfully complete a dump transaction or dump transaction with truncate_only command

Table 28-3: Special dump transaction options

Information provided	Task		
	Truncate log on same segment as data	**Truncate log without making a copy**	**Truncate log with insufficient free space**
Command	`dump transaction`	`dump transaction`	`dump transaction`
Database name	*`database_name`*	*`database_name`*	*`database_name`*
Do not copy log	`with truncate_only`	`with truncate_only`	`with no_log`

The remainder of this chapter provides greater detail about the information specified in dump and load commands and volume change messages. Routine dumps and loads are described first, followed by log dumps after device failure and the special syntax for truncating logs without making a backup copy.

For information about the permissions required to execute the dump and load commands, refer to “Designating responsibility for backups” on page 862.

Specifying the database and dump device

At a minimum, all dump and load commands must include the name of the database being dumped or loaded. Commands that dump or load data (rather than just truncating a transaction log) must also include a dump device.

Table 28-4 shows the syntax for backing up and loading a database or log.

Table 28-4: Indicating the database name and dump device

	Backing up a database or log	**Loading a database or log**
Database name Dump device	`dump {database	tran}` *`database_name`*	`load {database	tran}` *`database_name`*
	`to[compress::[`*`compression_ level`*`::]]`	`from [compress::]`
Dump device	*`stripe_device`*	*`stripe_device`*

Backing up a database or log	Loading a database or log

[at *server_name*]
[density = *density*,
blocksize = *number_bytes*,
capacity = *number_kilobytes*,
dumpvolume = *volume_name*,
file = *file_name*]
[stripe on
[compress::[*compression_level*
::]] *stripe_device*
[at *server_name*]
[density = *density*,
blocksize = *number_bytes*,
capacity = *number_kilobytes*,
dumpvolume = *volume_name*,
file = *file_name*] ...]
[with{
density = *density*,
blocksize = *number_bytes*,
capacity = *number_kilobytes*,
dumpvolume = *volume_name*,
file = *file_name*,
[nodismount | dismount],
[nounload | unload],
retaindays = *number_days*,
[noinit | init],
[notify = {client |
operator_console}]
standby_access}]
	[at *server_name*]
	[density = *density*,
	dumpvolume = *volume_name*
	file = *file_name*]
	[stripe on
	[compress::]*stripe_device*
	[at *server_name*]
	[density = *density*,
	dumpvolume = *volume_name*,
	file = *file_name*] ...]
	[with{
	density = *density*,
	dumpvolume = *volume_name*,
	file = *file_name*,
	[nodismount | dismount],
	[nounload | unload],
	[notify = {client |
	operator_console}]}]

Rules for specifying database names

You can specify the database name as a literal, a local variable, or a parameter to a stored procedure.

If you are loading a database from a dump:

- The database must exist. You can create a database with the for load option of create database, or load it over an existing database. Loading a database always overwrites all the information in the existing database.

- You do not need to use the same database name as the name of the database you dumped. For example, you can dump the pubs2 database, create another database called pubs2_archive, and load the dump into the new database.

Warning! You should never change the name of a database that contains primary keys for references from other databases. If you must load a dump from such a database and provide a different name, first drop the references to it from other databases.

Rules for specifying dump devices

When you specify a dump device:

- You can specify the dump device as a literal, a local variable, or a parameter to a stored procedure.
- You cannot dump to or load from the "null device" (on UNIX, */dev/null*; not applicable to PC platforms).
- When dumping to or loading from a local device, you can use any of the following forms to specify the dump device:
  - An absolute path name
  - A relative path name
  - A logical device name from the sysdevices system table

  The Backup Server resolves relative path names using Adaptive Server's current working directory.
- When dumping or loading over the network:
  - You must specify the absolute path name of the dump device. You cannot use a relative path name or a logical device name from the sysdevices system table.
  - The path name must be valid on the machine on which the Backup Server is running.
  - If the name includes any characters except letters, numbers, or the underscore (_), you must enclose it in quotes.
- If you dump a transaction log using with standby_access, you must load the dump using with standby_access.

Examples

The following examples use a single tape device for dumps and loads. (It is not necessary to use the same device for dumps and loads.)

- On UNIX:

```
dump database pubs2 to "/dev/nrmt4"
load database pubs2 from "/dev/nrmt4"
```

- On Windows NT:

```
dump database pubs2 to "\\.\tape0"
load database pubs2 from "\\.\tape0"
```

You can also dump to an operating system file. The following example is for Windows NT:

```
dump database pubs2 to "d:\backups\backup1.dat"
load database pubs2 from "d:\backupbackup1.dat"
```

Tape device determination by backup server

When you issue a dump database or dump transaction command, Backup Server checks whether the device type of the specified dump device is known (supplied and supported internally) by Adaptive Server. If the device is not a known type, Backup Server checks the tape configuration file (default location is *$SYBASE/backup_tape.cfg)* for the device configuration.

If the configuration is found, the dump command proceeds.

If the configuration is not found in the tape device configuration file, the dump command fails with the following error message:

```
Device not found in configuration file. INIT needs
to be specified to configure the device.
```

To configure the device, issue the dump database or dump transaction with the init parameter. Using operating system calls, Backup Server attempts to determine the device's characteristics; if successful, it stores the device characteristics in the tape configuration file.

If Backup Server cannot determine the dump device characteristics, it defaults to one dump per tape. The device cannot be used if the configuration fails to write at least one dump file.

Tape configuration by Backup Server applies only to UNIX platforms.

Tape sevice configuration file

Format

The tape device configuration file contains tape device information that is used only by the dump command.

The format of the file is one tape device entry per line. Fields are separated by blanks or tabs.

Creation

This file is created only when Backup Server is ready to write to it (dump database or dump transaction with init). When Backup Server tries to write to this file for the first time, the following warning message is issued:

```
Warning, unable to open device configuration file
for reading. Operating system error. No such file or
directory.
```

Ignore this message. Backup Server gives this warning and then creates the file and writes the configuration information to it.

Manual editing

The only user interaction with the file occurs when the user receives the following error message:

```
Device does not match the current configuration.
Please reconfigure this tape device by removing the
configuration file entry and issuing a dump with the
INIT qualifier.
```

This means that the tape hardware configuration changed for a device name. Delete the line entry for that device name and issue a dump command, as instructed.

Default location

The default path name for the configuration file is *$SYBASE/backup_tape.cfg*. You can change the default location with the Sybase installation utilities. See the installation documentation for your platform for more information.

Specifying the compress option

The dump command includes a compress option that allows you to compress databases and transaction logs using Backup Server.

Table 28-5 The shows the syntax for dump database ... compress and dump transaction ... compress commands is:

Table 28-5: Indicating the database name and dump device

	Backing up a database or log	**Loading a database or log**
	`dump {database | tran}` *`database_name`* `to`	`load {database | tran}` *`database_name`* `from`
Compress option	`[compress::[`*`compression_level`*`::]]`	`[compress::]`
	`stripe_device` `[at` *`server_name`*`] [density =` *`density`*`, blocksize =` *`number_bytes`*`, capacity =` *`number_kilobytes`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`] [stripe on`	*`stripe_device`*`} [at` *`server_name`*`] [density =` *`density`*`, dumpvolume =` *`volume_name`* `file =` *`file_name`*`] [stripe on`
Compress option	`[compress::[`*`compression_level`*`::]]`	`[compress::]`
	`stripe_device` `[at` *`server_name`*`] [density =` *`density`*`, blocksize =` *`number_bytes`*`, capacity =` *`number_kilobytes`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`] ...] [with{ density =` *`density`*`, blocksize =` *`number_bytes`*`, capacity =` *`number_kilobytes`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`, [nodismount | dismount], [nounload | unload], retaindays =` *`number_days`*`, [noinit | init], [notify = {client | operator_console}] standby_access}]`	*`stripe_device`* `[at` *`server_name`*`] [density =` *`density`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`] ...] [with{ density =` *`density`*`, dumpvolume =` *`volume_name`*`, file =` *`file_name`*`, [nodismount | dismount], [nounload | unload], [notify = {client | operator_console}]}]`

Syntax

The partial syntax specific to dump database … compress and dump transaction … compress is:

dump database *database_name*
to compress::[*compression_level*::]*stripe_device*
…[stripe on compress::[*compression_level*::]*stripe_device*] …

dump transaction *database_name*
to compress::[*compression_level*::]*stripe_device*
…[stripe on compress::[*compression_level*::]*stripe_device*]…

Where *database_name* is the database you are loading into, and compress::*compression_level* is a number between 0 and 9, with 0 indicating no compression, and 9 providing the highest level of compression. If you do not specify *compression_level*, the default is 1. *stripe_device* is the full path to the archive file of the database or transaction log you are compressing. If you do not include a full path for your dump file, Adaptive Server creates a dump file in the directory in which you started Adaptive Server.

Use the stripe on clause to use multiple dump devices for a single dump. See "Specifying additional dump devices: the stripe on clause" on page 906 for more information about the stripe on clause.

Note The compress option works only with local archives; you cannot use the servername option.

Example

```
dump database pubs2 to
    "compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp"
```

```
Backup Server session id is:  9.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.132.1.1: Attempting to open byte stream device:
'compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp::00'
Backup Server: 6.28.1.1: Dumpfile name 'pubs2002580BD27  ' section number 1
mounted on byte stream
'compress::4::/opt/bin/Sybase/dumps/dmp090100.dmp::00'
Backup Server: 4.58.1.1: Database pubs2: 394 kilobytes DUMPed.
Backup Server: 4.58.1.1: Database pubs2: 614 kilobytes DUMPed.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database pubs2: 622 kilobytes DUMPed.
Backup Server: 3.42.1.1: DUMP is complete (database pubs2).
```

The *compression_level* must be a number between 0 and 9. The compress option does not recognize numbers outside this range, and treats them as part of the file name while it compresses your files using the default compression level. For example, the following syntax creates a file called *99::pubs2.cmp*, which is compressed with the default compression level of 1:

```
dump database pubs2 to "compress::99::pubs2.cmp"
```

In general, the higher the compression numbers, the smaller your archives are compressed into. However, the compression result depends on the actual content of your files.

Table 28-6 shows the compression levels for the pubs2 database. These numbers are for reference only; the numbers for your site may differ depending on OS level and configuration.

Table 28-6: Compression levels and compressed file sizes for pub2

Compression levels	Compressed file size
No compression/Level 0	630K
Default compression/Level 1	128K
Level 2	124K
Level 3	121K
Level 4	116K
Level 5	113K
Level 6	112K
Level 7	111K
Level 8	110K
Level 9	109K

The higher the compression level, the more CPU-intensive the process is.

For example, you may not want to use a level-9 compression when archiving your files. Instead, consider the trade-off between processing effort and archive size. Compression level 6 provides optimal CPU usage, producing an archive that is 60 percent to 80 percent smaller than a regular uncompressed archive. Sybase recommends that you initially use compression level 6, then increase or decrease the level based on your performance requirements.

For complete syntax information about dump database and dump transaction, see the *Reference Manual*.

Backup Server dump files and compressed dumps

When you perform dump database or dump transaction using an archive file that already exists, Backup Server automatically checks the header of the existing dump archive. If the header is unreadable, Backup Server assumes that the file is a valid non-archive file, and prompts you to change the dump archive with the following message:

```
Backup Server: 6.52.1.1: OPERATOR: Volume to be overwritten on
'/opt/SYBASE/DUMPS/model.dmp' has unrecognized label data.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
       @session_id = 5,
       @devname = '/opt/SYBASE/DUMPS/model.dmp',
       @action = { 'PROCEED' | 'RETRY' |
'ABORT' },
       @vname = <new_volume_name>
```

For this reason, if you perform dump database or dump transaction to a file without the compress:: option into an existing compressed dump archive, Backup Server does not recognize the header information of the archive because it is compressed.

Example

The second dump database reports an error, and prompts you with sp_volchanged:

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
go
```

To prevent this error, either:

- Include the with init option in your subsequent dump database and dump transaction commands:

  ```
  dump database model to 'compress::model.cmp'
  go
  dump database model to 'model.cmp'
     with init
  go
  ```

- Include the compress:: option in subsequent dump database and dump transaction commands:

  ```
  dump database model to 'compress::model.cmp'
  go
  dump database model to 'compress::model.cmp'
  go
  ```

Using the compress:: option into uncompressed dump archives, as in this example, does not generate errors:

```
dump database model to 'model.cmp'
go
dump database model to 'compress::model.cmp'
go
```

Loading databases and transaction logs dumped with compress option

If you use dump ... compress to dump a database or transaction log, you must load this dump using the load ... compress option.

The partial syntax for load database ... compress and load transaction ... compress is:

```
load database database_name
from compress::stripe_device
…[stripe on compress::stripe_device]…

load transaction database_name
from compress::stripe_device
…[stripe on compress::stripe_device]…
```

The *database_name* in the syntax is the database you archived, and compress:: invokes the decompression of the archived database or transaction log. *archive_name* is the full path to the archived database or transaction log that you are loading. If you did not include a full path when you created your dump file, Adaptive Server created a dump file in the directory in which you started Adaptive Server.

Use the stripe on clause if you compressed the database or transaction log using multiple dump. See "Specifying additional dump devices: the stripe on clause" on page 906 for more information about the stripe on clause.

Note Do not use the *compression_level* variable for the load command.

Example

```
load database pubs2 from
    "compress::/opt/bin/Sybase/dumps/dmp090100.dmp"

Backup Server session id is:  19.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.132.1.1: Attempting to open byte stream device:
'compress::/opt/bin/Sybase/dumps/dmp090100.dmp::00'
Backup Server: 6.28.1.1: Dumpfile name 'pubs2002620A951  ' section number 1
mounted on byte stream 'compress::/opt/bin/Sybase/dumps/dmp090100.dmp::00'
Backup Server: 4.58.1.1: Database pubs2: 1382 kilobytes LOADed.
Backup Server: 4.58.1.1: Database pubs2: 3078 kilobytes LOADed.
Backup Server: 4.58.1.1: Database pubs2: 3086 kilobytes LOADed.
Backup Server: 3.42.1.1: LOAD is complete (database pubs2).
Use the ONLINE DATABASE command to bring this database online; SQL Server
will not bring it online automatically.
```

For complete syntax information about load database and load transaction, see the *Reference Manual*.

Specifying a remote Backup Server

Use the at *server_name* clause to send dump and load requests over the network to a Backup Server running on another machine.

Table 28-7 shows the syntax for dumping or loading from a remote Backup Server.

Table 28-7: Dumping to or loading from a remote Backup Server

	Backing up a database or log	Loading a database or log
	`dump {database | tran}` *`database_name`*	`load {database | tran}` *`database_name`*
Remote Backup Server	`[at` *`server_name`*`]`	`[at` *`server_name`*`]`
	`[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`]` `[stripe on` `[at` *`server_name`*`]` `[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]` `[with{` `density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`,` `[nodismount | dismount],` `[nounload | unload],` `retaindays =` *`number_days`*`,` `[noinit | init],` `[notify = {client |` `operator_console}]` `standby_access}]`	`[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`]` `[stripe on` `[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]` `[with{` `density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`,` `[nodismount | dismount],` `[nounload | unload],` `[notify = {client |` `operator_console}]`

Note The compress option works only with local archives; you cannot use the *backup_server_name* option.

Sending dump and load requests over the network is ideal for installations that use a single machine with multiple tape devices for all backups and loads. Operators can be stationed at these machines, ready to service all tape change requests.

The following examples dump to and load from the remote Backup Server REMOTE_BKP_SERVER:

```
dump database pubs2 to "/dev/nrmt0" at REMOTE_BKP_SERVER
load database pubs2 from "/dev/nrmt0" at REMOTE_BKP_SERVER
```

The *server_name* must appear in the interfaces file on the computer where Adaptive Server is running, but does not need to appear in the sysservers table. The *server_name* must be the same in both the local and the remote interfaces file.

Specifying tape density, block size, and capacity

In most cases, the Backup Server uses a default tape density and block size that are optimal for your operating system; *we recommend that you use them*.

You can specify a density, block size, and capacity for each dump device. You can also specify the density, blocksize, and capacity options in the with clause for all dump devices. Characteristics that are specified for an individual tape device take precedence over those that you specify using the with clause.

Table 28-8 shows the syntax for specifying the tape density, block size, and capacity.

Table 28-8: Specifying tape density, block size, and capacity

Backing up a database or log	Loading a database or log
`dump {database | tran}` *database_name* `to [compress::[`*compression_ level*`::]]` *stripe_device* `[at` *server_name*`]`	`load {database | tran}` *database_name* `from [compress::]`*stripe_device* `[at` *server_name*`]`

	Backing up a database or log	Loading a database or log

Characteristics of a single tape device	`[density = `*`density`*`,`
	`blocksize = `*`number_bytes`*`,`
	`capacity = `*`number_kilobytes`*`,`
		`[density = `*`density`*`,`

	`dumpvolume = `*`volume_name`*`,`
	`file = `*`file_name`*`]`
	`[stripe on`
	`[compress::[`*`compression_`*
	`level``::]] `*`stripe_device`*
	`[at `*`server_name`*`]`
	`[density = `*`density`*`,`
	`blocksize = `*`number_bytes`*`,`
	`capacity = `*`number_kilobytes`*`,`
	`dumpvolume = `*`volume_name`*`,`
	`file = `*`file_name`*`] ...]`
		`dumpvolume = `*`volume_name`*`,`
		`file = `*`file_name`*`]`
		`[stripe on`
		`[compress::]`*`stripe_device`*
		`[at `*`server_name`*`]`
		`[density = `*`density`*`,`
		`dumpvolume = `*`volume_name`*`,`
		`file = `*`file_name`*`] ...]`

Characteristics of all dump devices	`[with{`
	`density = `*`density`*`,`
	`blocksize = `*`number_bytes`*`,`
	`capacity = `*`number_kilobytes`*`,`
		`[with{`
		`density = `*`density`*`,`

	`dumpvolume = `*`volume_name`*`,`
	`file = `*`file_name`*`,`
	`[nodismount | dismount],`
	`[nounload | unload],`
	`retaindays = `*`number_days`*`,`
	`[noinit | init],`
	`[notify = {client |`
	`operator_console}]`
	`standby_access}]`
		`dumpvolume = `*`volume_name`*`,`
		`file = `*`file_name`*`,`
		`[nodismount | dismount],`
		`[nounload | unload],`
		`[notify = {client |`
		`operator_console}]`

The following sections provide greater detail about the density, blocksize, and capacity options.

Overriding the default density

The dump and load commands use the default tape density for your operating system. In most cases, this is the optimal density for tape dumps.

This option has no effect on UNIX and PC platform dumps or loads.

Note Specify tape density only when using the init tape handling option. For more information on this option, see “Reinitializing a volume before a dump” on page 911.

Overriding the default block size

The blocksize parameter specifies the number of bytes per I/O operation for a dump device. By default, the dump and load commands choose the "best" block size for your operating system. Wherever possible, use these defaults.

You can use the blocksize = *number_bytes* option to override the default block size for a particular dump device. The block size must be at least one database page (2048 bytes) and must be an exact multiple of the database page size.

For UNIX systems, the block size specified on the load command is ignored. Backup Server uses the block size that was used to make the dump.

Specifying a higher block size value

If you dump to a tape using the dump database or dump transaction commands, and specify a block size value which is higher than the maximum blocksize of a device as determined by Backup Server, then the dump or the load may fail on certain tape drives. An operating system error message displays; for example, on an 8mm tape drive on HP the error message is:

```
Backup Server: 4.141.2.22: [2] The 'write' call
failed for device 'xxx' with error number 22 (Invalid
argument). Refer to your operating system
documentation for further details.
```

You should not specify a block size greater than the device's block size stored in the tape device configuration file in *$SYBASE/backup_tape.cfg*. The block size for a device is the fifth field of the line in the tape device configuration file.

This error occurs only on tape drives where tape auto config is run; that is, the device models are not hard-coded in Backup Server code.

Specifying tape capacity for dump commands

For UNIX platforms that cannot reliably detect the end-of-tape marker, you must indicate how many kilobytes can be dumped to a tape.

If you specify the physical path name of the dump device, you must include the capacity = *number_kilobytes* parameter in the dump command. If you specify the logical dump device name, the Backup Server uses the *size* parameter stored in the sysdevices table, unless you override it with the capacity = *number_kilobytes* parameter.

The specified capacity must be at least five database pages (each page requires 2048 bytes). We recommend that you specify a capacity that is slightly below the capacity rated for your device.

A general rule for calculating capacity is to use 70 percent of the manufacturer's maximum capacity for the device, and allow 30 percent for overhead (inter-record gaps, tape marks, and so on). This rule works in most cases, but may not work in all cases because of differences in overhead across vendors and devices.

Non-rewinding tape functionality for Backup Server

The non-rewinding tape functionality automatically positions the tape at the end of valid dump data, which saves time when you want to perform multiple dump operations.

Dump label changes

Backup Server writes an End-of-File label, EOF3, at the end of every dump operation.

Note You cannot load a tape with this label into any version of Adaptive Server earlier then 12.0.

Tape operations

When a new dump is performed, Backup Server performs a scan for the last EOF3 label.

If the EOF3 label is found, the pertinent information is saved and the tape is positioned forward to the beginning of the next file on tape. This is the new append point.

If the EOF3 label is not found or any other problem is encountered, Backup Server rewinds the tape and scans forward. Any error that occurs during these steps does not abort the dump operation, but causes Backup Server to default to rewind-and-scan behavior. If the error persists during the rewind and scan, the dump command aborts.

Dump version compatibility

Backup Server activates non-rewinding logic only if the label version on the tape is greater than or equal to 5. Therefore, a dump command with the with init clause is needed to activate this logic. If a dump without init is initiated onto a volume with a label version less than 5, you are prompted to change the volume, and the dump starts on the next volume. The label version of a multi-volume dump does not change in the middle of the volume set.

Table 28-9 defines the label versions for which the new behavior is enabled.

Table 28-9: Label version compatibility

Label version	Enabled
‘3’	No
‘4’	No
‘5’	Yes
‘6’	Yes

Specifying the volume name

Use the with dumpvolume = *volume_name* option to specify the volume name. dump database and dump transaction write the volume name to the SQL tape label. load database and load transaction check the label. If the wrong volume is loaded, Backup Server generates an error message.

You can specify a volume name for each dump device. You can also specify a volume name in the with clause for all devices. Volume names specified for individual devices take precedence over those specified in the with clause.

Table 28-10 shows the syntax for specifying a volume name.

Table 28-10: Specifying the volume name

	Backing up a database or log	**Loading a database or log**
	dump {database | tran} *database_name* to [compress::[*compression_ level*::]] *stripe_device* [at *server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,	load {database | tran} *database_name* from [compress::]*stripe_ device* [at *server_name*] [density = *density*,
Volume name for single device	dumpvolume = *volume_name*,	dumpvolume = *volume_name*,
	file = *file_name*] [stripe on [compress::[*compression_ level*::]] *stripe_device* [at *server_name*] [density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*, dumpvolume = *volume_name*, file = *file_name*] ...] [with{ density = *density*, blocksize = *number_bytes*, capacity = *number_kilobytes*,	file = *file_name*] [stripe on [compress::]*stripe_ device* [at *server_name*] [density = *density*, dumpvolume = *volume_name*, file = *file_name*]...] [with { density = *density*,
Volume name for all devices	dumpvolume = *volume_name*,	dumpvolume = *volume_name*,
	file = *file_name*, [nodismount | dismount], [nounload | unload], retaindays = *number_days*, [noinit | init], [notify = {client |operator_console}] standby_access}]	file = *file_name*, [nodismount | dismount], [nounload | unload], [notify = {client | operator_console}]

Loading from a multifile volume

When you load a database dump from a volume that contains multiple dump files, specify the dump file name. If you omit the dump file name and specify only the database name, Backup Server loads the first dump file into the specified database. For example, entering the following command loads the first dump file from the tape into pubs2, regardless of whether that dump file contains data from pubs2:

```
load database pubs2 from "/dev/rdsk/clt3d0s6"
```

To avoid this problem, specify a unique dump file name each time you dump or load data. To get information about the dump files on a given tape, use the listonly = full option of load database.

Identifying a dump

When you dump a database or transaction log, Backup Server creates a default file name for the dump by concatenating the:

- Last 7 characters of the database name
- 2-digit year number
- 3-digit day of the year (1–366)
- Number of seconds since midnight, in hexadecimal

You can override this default using the file = *file_name* option. The file name cannot exceed 17 characters and must conform to the file naming conventions for your operating system.

You can specify a file name for each dump device. You can also specify a file name for all devices in the with clause. File names specified for individual devices take precedence over those specified in the with clause.

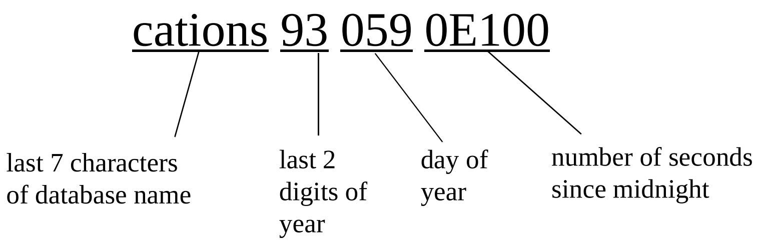
Table 28-11 shows the syntax for specifying the name of a dump.

Table 28-11: Specifying the file name for a dump

	Backing up a database or log	Loading a database or log
	`dump {database | tran}` *`database_name`* `to` `[compress::[`*`compression_level`*`::]]` *`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,`	`load {database | tran}` *`database_name`* `from` `[compress::]`*`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,`
File name for single device	`file =` *`file_name`*`]`	`file =` *`file_name`*`]`
	`[stripe on` `[compress::[`*`compression_level`*`::]]` *`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]` `[with{` `density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,`	`[stripe on` `[compress::]`*`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]` `[with{` `density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,`
File name for all devices	`file =` *`file_name`*`,`	`file =` *`file_name`*`,`
	`[nodismount | dismount],` `[nounload | unload],` `retaindays =` *`number_days`*`,` `[noinit | init],` `[notify = {client |` `operator_console}]` `standby_access}]`	`[nodismount | dismount],` `[nounload | unload],` `[notify = {client` `| operator_console}]`

The following examples dump the transaction log for the publications database without specifying a file name. The default file name, *cations930590E100*, identifies the database and the date and time the dump was made:

Figure 28-1: File-naming convention for database and transaction log dumps


cations 93 059 0E100
last 7 characters of database name
last 2 digits of year
day of year
number of seconds since midnight

Backup Server sends the file name to the default message destination or to the notify location for the dump command. Be sure to label each backup tape with the volume name and file name before storing it.

When you load a database or transaction log, you can use the file = *file_name* clause to specify which dump to load from a volume that contains multiple dumps.

When loading the dump from a multifile volume, you must specify the correct file name.

```
dump tran publications
    to "/dev/nrmt3"
load tran publications
    from "/dev/nrmt4"
    with file = "cations930590E100"
```

The following examples use a user-defined file-naming convention. The 15-character file name, *mydb97jul141800*, identifies the database (mydb), the date (July 14, 1997), and the time (18:00, or 6:00 p.m.) that the dump was made. Using the load command advances the tape to *mydb97jul141800* before loading:

```
dump database mydb
    to "/dev/nrmt3"
    with file = "mydb97jul141800"
load database mydb
    from "/dev/nrmt4"
    with file = "mydb97jul141800"
```

Improving dump or load performance

When you start Backup Server, you can use the -m parameter to improve the performance of the dump and load commands by configuring more shared memory for the Backup Server. The -m parameter specifies the maximum amount of shared memory used by the Backup Server. You must also configure your operating system to ensure that this amount of shared memory is available to the Backup Server. After a dump or load operation is completed, its shared memory segments are released.

Note Configuring more shared memory improves dump/load performance only if the performance limits of the hardware setup have not been reached. Increasing the value of -m may not result in improved performance when dumping to a slow tape device such as QIC, but it can improve performance significantly when dumping to a faster device, such as DLT.

Compatibility with prior versions

There are some compatibility issues between dump files and Backup Server. Table 28-12 indicates the dump file formats that can be loaded by the current and previous versions of local Backup Servers.

Table 28-12: Server for local operations

	New dump file format	**Old dump file format**
New version of server	Yes	Yes
Prior version of server	No	Yes

Table 28-13 and Table 28-14 indicate the dump file formats that can be loaded by the current and prior versions of remote Backup Servers. In a remote Backup Server scenario, the master server is the Backup Server on the same machine as the database and Adaptive Server Enterprise, and the slave server is the Backup Server on the same remote machine as the archive device.

Table 28-13 indicates the load operations that work when master server is the current version of Backup Server.

Table 28-13: New version of master server

	New dump file format	**Old dump file format**
New slave version of server	Yes	Yes

	New dump file format	**Old dump file format**
Prior slave version of server	No	Yes

Table 28-14 indicates the load operations that work when the master server is a prior version.

Table 28-14: Prior version of master server

	New dump file format	**Old dump file format**
New slave version of server	No	Yes
Prior slave version of server	No	Yes

Labels stored in integer format

Backup Server 12.0 and later store the stripe number in integer format. Earlier versions of Backup Server stored the 4-byte stripe number in the HDR1 label in ASCII format. These earlier versions of Backup Server cannot load a dump file that uses the newer dump format. However, Backup Server version 12.0 and later can read and write earlier versions of the dump format.

When performing a dump or load operation involving one or more remote servers, the operation aborts with an error message, if:

- The versions of one or more of the remote Backup Servers are earlier than 12.0, and the database is dumped to or loaded from more than 32 stripes.

  Or:

- The dump file from which one or more of the Backup Servers are reading during a load is from an earlier version's format, and the number of stripes from which the database is loaded is greater than 32.

Configuring system resources

Before you perform dumps and loads. you must configure the local and remote Backup Servers at start-up by providing the appropriate values for the system resources controlled by the command-line options. See the *Utility Guide* for a complete list of the command-line options.

If your system resources are not configured properly, the dump or load can fail. For example, a remote dump to greater than 25 stripes with the local and remote Backup Servers started with default configuration will fail because the maximum number of network connections that Backup Server can originate (specified by the -N option) is 25; however, by default the maximum number of server connections into the remote Backup Server (specified by the -C option) is 30.

To configure the system to use the higher stripe limitations, set the following operating system parameters:

- Number of shared memory segments to which a process can attach.
- Number of shared memory identifiers
- Swap space

If these parameters are not configured properly, when a dump is started to (or a load is started from) a large number of stripes, the operation may abort because of lack of system resources. In this case, you receive a message that Backup Server could not create or attach to a shared memory segment and therefore the SYBMULTBUF processes are terminated.

Setting shared memory usage

The syntax for starting Backup Server with the -m parameter is, where *nnn* is the maximum amount of shared memory in megabytes that the Backup Server can use for all of its dump or load sessions:

```
backupserver [-m nnn]
```

The -m parameter sets the upper limit for shared memory usage. However, Backup Server may use less memory than specified if it detects that adding more memory will not improve performance.

Backup Server determines the amount of shared memory available for each stripe by dividing the -m value by the configured number of service threads (-P parameter).

The default value for -m is the number of service threads multiplied by 1MB. The default value for -P is 48, so the default maximum shared memory utilization is 48MB. However, Backup Server reaches this usage only if all the 48 service threads are active concurrently. The maximum value for -P is the maximum number of service threads, 12,288. (For more information about -P, see “Configuring user-defined roles” on page 358.)

The amount of shared memory per stripe available for Backup Server is inversely proportional to the number of service threads you allocate. If you increase the maximum number of service threads, you must increase the -m value, also, to maintain the same amount of shared memory per stripe. If you increase the -P value but do not increase the -m value, the shared memory allocated per stripe can decrease to the point that the dump or load cannot be processed.

To determine how much to increase the -m value, use this formula:

(-m value in MB) * 1024/(-P value)

If the value obtained by this formula is less than 128KB, Backup Server will not start.

The minimum value for -m is 6MB. The maximum value for -m depends on operating system limits on shared memory.

If you create a dump using a Backup Server with a high shared memory value, and attempt to load the dump using a Backup Server with a lower shared memory value, Backup Server uses only the available memory. This results in degradation of the load performance.

If the amount of shared memory available per stripe at load time is less than twice the block size used at dump time, Backup Server aborts the load with an error message.

Setting maximum number of stripes

The maximum number of stripes that Backup Server can use is limited by the maximum number of Open Server threads it can create. Open Server imposes a maximum limit of 12k on the number of threads an application can create.

Backup Server creates one service thread for each stripe. Therefore, the maximum number of local stripes Backup Server can dump to or load from is 12,286.

As an additional limitation, Backup Server uses two file descriptors for each stripe, apart from the file descriptors associated with the error log file, interfaces file, and other system files. However, there is a per-thread limitation imposed by the operating system on the number of file descriptors. Open Server has a limitation of 1280 on the number of file descriptors that an application can keep track of.

The formula for determining the approximate maximum number of local stripes to which Backup Server can dump is:

$$\frac{\textbf{(The smaller of either the OS limitation or the OpenServer limitation) - 2}}{\textbf{2}}$$

The formula for determining the approximate maximum number of remote stripes to which Backup Server can dump is:

$$\frac{\textbf{(The smaller of either the OS limitation or the OpenServer limitation) - 2}}{\textbf{3}}$$

For details about the default and maximum file descriptor limits, see your operating system documentation.

Setting maximum number of network connections

The maximum number of network connections a local Backup Server can originate is limited by Open Server to 9118. Because of this, the maximum number of remote stripes that Backup Server can use in a single dump or load operation is 9118.

A remote Backup Server accepts a maximum of 4096 server connections at any one time. Therefore, the maximum number of remote stripes to a single remote Backup Server is 4096.

Setting maximum number of service threads

The -P parameter for Backup Server configures the number of service threads Open Server creates. The maximum number of service threads is 12,228. The minimum values is 6. The maximum number of threads equals the maximum number of stripes available. If you have started Backup Server without setting a high enough -P value, and you attempt to dump or load a database to a number of stripes that exceeds the number of threads, the dump or load operation fails.

Specifying additional dump devices: the *stripe on* clause

Striping allows you to use multiple dump devices for a single dump or load command. Use a separate stripe on clause to specify the name (and, if desired, the characteristics) of each device.

Each dump or load command can have multiple stripe on clauses.

Table 28-15 shows the syntax for using more than one dump device.

Table 28-15: Using more than one dump device

	Backing up a database or log	Loading a database or log
	`dump {database | tran}` *`database_name`* `to [compress::[`*`compression_ level`*`::]]` *`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`]`	`load {database | tran}` *`database_name`* `from` `[compress::]`*`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`]`
Characteristics of an additional tape device (one set per device; up to 31 devices)	`[stripe on` `[compress::[`*`compression_ level`*`::]]` *`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]`	`[stripe on` `[compress::]`*`stripe_device`* `[at` *`server_name`*`]` `[density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`] ...]`

	Backing up a database or log	Loading a database or log
	`[with{` `density =` *`density`*`,` `blocksize =` *`number_bytes`*`,` `capacity =` *`number_kilobytes`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`,` `[nodismount | dismount],` `[nounload | unload],` `retaindays =` *`number_days,`* `[noinit | init],` `[notify = {client | operator_console}]` `standby_access}]`	`[with{` `density =` *`density`*`,` `dumpvolume =` *`volume_name`*`,` `file =` *`file_name`*`,` `[nodismount | dismount],` `[nounload | unload],` `[notify = {client | operator_console}]`

Dumping to multiple devices

The Backup Server divides the database into approximately equal portions and sends each portion to a different device. Dumps are made concurrently on all devices, reducing the time required to dump an individual database or transaction log. Because each tape stores only a portion of the database, it is less likely that a new tape will have to be mounted on a particular device.

Warning! Do not dump the master database to multiple tape devices. When loading the master database from tape or other removable media, you cannot change volumes unless you have another Adaptive Server that can respond to volume change messages.

Loading from multiple devices

You can use multiple devices to load a database or transaction log. Using multiple devices decreases both the time required for the load and the likelihood of having to mount multiple tapes on a particular device.

Using fewer devices to load than to dump

You can load a database or log even if one of your dump devices becomes unavailable between the dump and load. Specify fewer stripe clauses in the load command than you did in the dump command.

Note When you dump and load over the network, you must use the same number of drives for both operations.

The following examples use three devices to dump a database but only two to load it:

```
dump database pubs2 to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    stripe on "/dev/nrmt2"
load database pubs2 from "/dev/nrmt0"
    stripe on "/dev/nrmt1"
```

After the first two tapes are loaded, a message notifies the operator to load the third.

You can also dump a database to multiple operating system files. The following example is for Windows NT:

```
dump database pubs2 to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
load database pubs2 from "/dev/nrmt0"
stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
```

Specifying the characteristics of individual devices

Use a separate at *server_name* clause for each stripe device attached to a remote Backup Server. If you do not specify a remote Backup Server name, the local Backup Server looks for the dump device on the local machine. If necessary, you can also specify separate tape device characteristics (density, blocksize, capacity, dumpvolume, and file) for individual stripe devices.

The following examples use three dump devices, each attached to the remote Backup Server REMOTE_BKP_SERVER.

On UNIX:

```
dump database pubs2
    to "/dev/nrmt0" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt1" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt2" at REMOTE_BKP_SERVER
```

Tape handling options

The tape handling options, which appear in the with clause, apply to all devices used for the dump or load. They include:

- nodismount to keep the tape available for additional dumps or loads
- unload to rewind and unload the tape following the dump or load
- retaindays to protect files from being overwritten
- init to reinitialize the tape rather than appending the dump files after the last end-of-tape mark

Table 28-16 shows the syntax for tape handling options.

Table 28-16: Tape handling options

	Backing up a database or log	Loading a database or log
	`dump {database | tran}` *database_name* `to` `[compress::[`*compression_level*`::]]` *stripe_device* `[at` *server_name*`]` `[density =` *density*`,` `blocksize =` *number_bytes*`,` `capacity =` *number_kilobytes*`,` `dumpvolume =` *volume_name*`,` `file =` *file_name*`]` `[stripe on` `[compress::[`*compression_level*`::]]` *stripe_device* `[at` *server_name*`]` `[density =` *density*`,` `blocksize =` *number_bytes*`,` `capacity =` *number_kilobytes*`,` `dumpvolume =` *volume_name*`,` `file =` *file_name*`] ...]` `[with{` `density =` *density*`,` `blocksize =` *number_bytes*`,` `capacity =` *number_kilobytes*`,` `dumpvolume =` *volume_name*`,` `file =` *file_name*`,`	`load {database | tran}` *database_name* `from` `[compress::]`*stripe_device* `[at` *server_name*`]` `[density =` *density*`,` `dumpvolume =` *volume_name* `file =` *file_name*`]` `[compress::]`*stripe_device* `[at` *server_name*`]` `[density =` *density*`,` `dumpvolume =` *volume_name*`,` `file =` *file_name*`] ...]` `[with{` `density =` *density*`,` `dumpvolume =` *volume_name*`,` `file =` *file_name*`,`
Tape Handling Options	`[nodismount | dismount],` `[nounload | unload],` `retaindays =` *number_days*`,` `[noinit | init],`	`nodismount | dismount],` `[nounload | unload],`
	`[notify = {client |` `operator_console}]` `standby_access}]`	`[notify = {client |` `operator_console}]`

Specifying whether to dismount the tape

On platforms that support logical dismounts, tapes are dismounted when a dump or load completes. Use the nodismount option to keep the tape mounted and available for additional dumps or loads. This command has no effect on UNIX or PC systems.

Rewinding the tape

By default, both dump and load commands use the nounload tape handling option.

On UNIX systems, this prevents the tape from rewinding after the dump or load completes. This allows you to dump additional databases or logs to the same volume or to load additional databases or logs from that volume. Use the unload option for the last dump on the tape to rewind and unload the tape when the command completes.

Protecting dump files from being overwritten

tape retention in days specifies the number of days that must elapse between the creation of a tape file and the time at which you can overwrite it with another dump. This server-wide variable, which you can is set with sp_configure, applies to all dumps requested from a single Adaptive Server.

Use the retaindays = *number_days* option to override the tape retention in days parameter for a single database or transaction log dump. The number of days must be a positive integer, or zero if the tape can be overwritten immediately.

Note tape retention in days and retaindays are meaningful only for disk, 1/4-inch cartridge, and single-file media. On multifile media, Backup Server checks only the expiration date of the first file.

Reinitializing a volume before a dump

By default, each dump is appended to the tape following the last end-of-tape mark. Tape volumes are not reinitialized. This allows you to dump multiple databases to a single volume. (New dumps can be appended only to the last volume of a multivolume dump.)

Use the init option to overwrite any existing contents of the tape. If you specify init, the Backup Server reinitializes the tape *without* checking for:

- ANSI access restrictions
- Files that have not yet expired

- Non-Sybase data

The default, noinit, checks for all three conditions and sends a volume change prompt if any are present.

The following example initializes two devices, overwriting the existing contents with the new transaction log dumps:

```
dump transaction pubs2
    to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    with init
```

You can also use the init option to overwrite an existing file, if you are dumping a database to an operating system file. The following example is for Windows NT:

```
dump transaction pubs2
    to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    with init
```

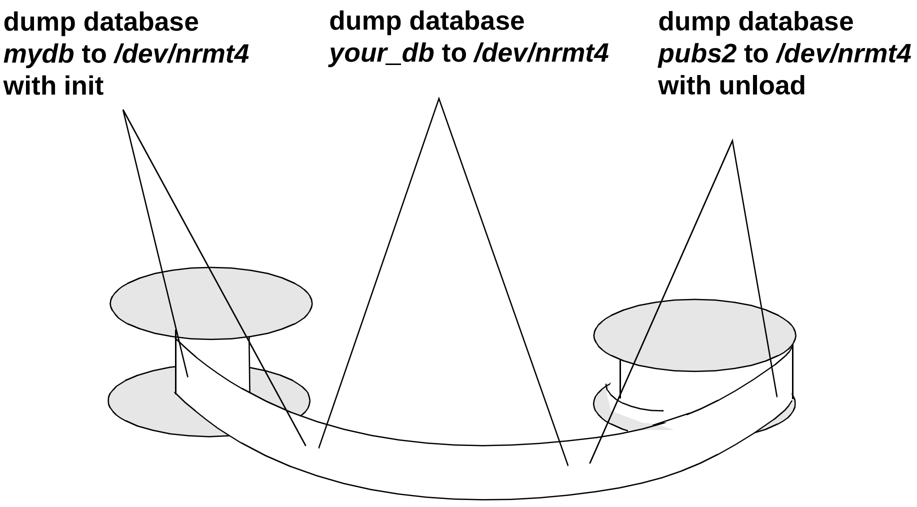
Dumping multiple databases to a single volume

To dump multiple databases to the same tape volume:

1. Use the init option for the first database. This overwrites any existing dumps and places the first dump at the beginning of the tape.
2. Use the default (noinit and nounload) option for subsequent databases. This places them one after the other on the tape.
3. Use the unload option for the last database on the tape. This rewinds and unloads the tape after you dump the last database.

Figure 28-2 illustrates how use to dump three databases to a single tape volume.

Figure 28-2: Dumping several databases to the same volume


dump database
mydb* to */dev/nrmt4
with init
dump database
your_db* to */dev/nrmt4
dump database
pubs2* to */dev/nrmt4
with unload

Overriding the default message destination

Backup Server messages inform the operator when to change tape volumes and how the dump or load is progressing. The default destination for these messages depends on whether the operating system offers an operator terminal feature.

The notify option, which appears in the with clause, allows you to override the default message destination for a dump or load. For this option to work, the controlling terminal or login session from which Backup Server was started must remain active for as long as Backup Server is working; otherwise, the sp_volchanged message is lost.

On operating systems that offer an operator terminal feature, volume change messages are always sent to an operator terminal on the machine where Backup Server is running. Use notify = client to route other Backup Server messages to the terminal session where the dump or load request initiated.

On systems such as UNIX that do not offer an operator terminal feature, messages are sent to the client that initiated the dump or load request. Use notify = operator_console to route messages to the terminal where the remote Backup Server was started.

Table 28-17 shows the syntax for overriding the default message destination.

Table 28-17: Overriding the default message destination

	Backing up a database or log	Loading a database or log

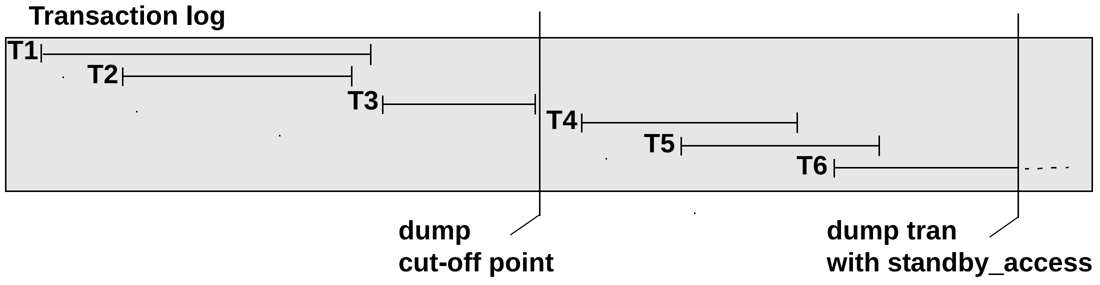
	dump {database | tran}
	database_name
	to
	[compress::[*compression_level*
	::]] *stripe_device*
	[at *server_name*]
	[density = *density,*
	blocksize = *number_bytes,*
	capacity = *number_kilobytes,*
	dumpvolume = *volume_name*,
	file = *file_name*]
	[stripe on
	[compress::[*compression_level*
	::]] *stripe_device*
	[at *server_name*]
	[density = *density*,
	blocksize = *number_bytes,*
	capacity = *number_kilobytes,*
	dumpvolume = *volume_name*,
	file = *file_name*] ...]
	[with{
	density = *density*,
	blocksize = *number_bytes*,
	capacity = *number_kilobytes*,
	dumpvolume = *volume_name*,
	file = *file_name*,
	[nodismount | dismount],
	[nounload | unload],
	retaindays = *number_days*,
	[noinit | init],
		load {database | tran}
		database_name
		from
		[compress::]*stripe_device*
		[at *server_name*]
		[density = *density*,
		dumpvolume = *volume_name*
		file = *file_name*]
		[stripe on
		[compress::]*stripe_device*
		[at *server_name*]
		[density = *density*,
		dumpvolume = *volume_name*,
		file = *file_name*] ...]
		[with{
		density = *density*,
		dumpvolume = *volume_name*,
		file = *file_name*,
		[nodismount | dismount],
		[nounload | unload],

Message destination	[notify = {client | operator_console}]	[notify = {client | operator_console}]
	standby_access}]	

Bringing databases online *with standby_access*

with standby_access causes dump transaction to dump only completed transactions. It dumps the transaction log up to the point at which there are no active transactions. If you do not use with standby_access, the entire transaction log, including records for all open transactions is dumped. A transaction log dump using with standby_access is illustrated in Figure 28-3.

Figure 28-3: Dump cut-off point for dump transaction with standby_access



In Figure 28-3, a dump transaction...with standby_access command is issued at a point where transactions T1 through T5 have completed and transaction T6 is still open. The dump cannot include T5 because T6 is still open, and it cannot include T4, because T5 is still open. Thus, the dump must stop at the end of transaction T3, where it will include completed transactions T1 through T3.

Syntax

The syntax for with standby_access is:

```
dump tran[saction] database_name to...
    [with standby_access]
```

For more information about the with dump tran...with standby_access option, see the *Reference Manual*.

When do I use *with standby_access*?

Use dump tran[saction]...with standby_access when you will be loading two or more transaction logs in sequence, and you want the database to be online between loads. For example, if you have a read-only database that gets its data by loading transaction dumps from a primary database. In this case, if the read-only database is used for generating a daily report based on transactions in the primary database, and the primary database's transaction log is dumped at the end of day, the daily cycle of operations is:

1. On the primary database: dump tran[saction]...with standby_access
2. On the read-only database: load tran[saction]...
3. On the read-only database: online database for standby_access

Warning! If a transaction log contains open transactions, and you dump it without using with standby_access, Adaptive Server will not allow you to load the log, bring the database online, and then load a subsequent transaction dump. If you are going to load a series of transaction dumps, you can bring the database online only after loading a dump originally made with standby_access or after loading the entire series.

Bring databases online *with standby_access*

The online database command also includes a with standby_access option. Use for standby_access to bring a database online after loading it with a dump that was made using the with standby_access option.

Warning! If you try to use online database for standby_access with a transaction log that was not dumped using the with standby_access option, the command will fail.

Syntax

The syntax for online database is:

```
online database database_name [for standby_access]
```

For more information about the with online database...for standby_access option, see the *Reference Manual*.

Getting information about dump files

If you are unsure of the contents of a tape, use the with headeronly or with listonly option of the load commands to request that information.

Table 28-18 shows the syntax for finding the contents of a tape.

Table 28-18: Listing dump headers or file names

	Listing information about a dump

	load {database | tran}
	database_name
	from [compress::]*stripe_device*
	[at *server_name*]
	[density = *density*,
	dumpvolume = *volume_name*
	file = *file_name*]
	[stripe on
	[compress::]*stripe_device*
	[at *server_name*]
	[density = *density*,
	dumpvolume = *volume_name*,
	file = *file_name*] ...]
	[with{
	density = *density*,
	dumpvolume = *volume_name*,
	file = *file_name*,
	[nodismount | dismount],
	[nounload | unload],

List header information List files on tape	[headeronly [, file = *filename*]], [listonly [= full]],
	[notify = {client | operator_console}]

Note Neither with headeronly nor with listonly loads the dump files after displaying the report.

Requesting dump header information

with headeronly returns the header information for a single file. If you do not specify a file name, with headeronly returns information about the first file on the tape.

The header indicates whether the dump is for a database or transaction log, the database ID, the file name, and the date the dump was made. For database dumps, it also shows the character set, sort order, page count, and next object ID. For transaction log dumps, it shows the checkpoint location in the log, the location of the oldest begin transaction record, and the old and new sequence dates.

The following example returns header information for the first file on the tape and then for the file *mydb9229510945*:

```
load database mydb
    from "/dev/nrmt4"
    with headeronly
load database mydb
    from "/dev/nrmt4"
    with headeronly, file = "mydb9229510945"
```

Here is sample output from headeronly:

```
Backup Server session id is: 44. Use this value when executing the
‘sp_volchanged’ system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 4.28.1.1: Dumpfile name ‘mydb9232610BC8 ’ section number 0001
mounted on device ‘backup/SQL_SERVER/mydb.db.dump’
This is a database dump of database ID 5 from Nov 21 1992 7:02PM.
Database contains 1536 pages; checkpoint RID=(Rid pageid = 0x404; row num =
0xa); next object ID=3031; sort order ID=50, status=0; charset ID=1.
```

Determining the database, device, file name, and date

with listonly returns a brief description of each dump file on a volume. It includes the name of the database, the device used to make the dump, the file name, the date and time the dump was made, and the date and time it can be overwritten. with listonly = full provides greater detail. Both reports are sorted by SQL tape label.

Following is sample output of a load database command with listonly:

```
Backup Server: 4.36.1.1: Device ‘/dev/nrst0’:
File name: ‘model9320715138  ’
Create date & time: Monday, Jul 26, 1993, 23:58:48
Expiration date & time:  Monday, Jul 26, 1993, 00:00:00
Database name: ‘model                         ’
```

The following is sample output from with listonly = full:

```
Backup Server: 4.37.1.1: Device ‘/dev/nrst0’:
```

```
Label id: 'HDR1'
File name:'model9320715138  '
Stripe count:0001
Device typecount:01
Archive volume number:0001
Stripe position:0000
Generation number:0001
Generation version:00
Create date & time:Monday, Jul 26, 1993, 23:58:48
Expiration date & time:Monday, Jul 26, 1993, 00:00:00
Access code:' '
File block count:000000
Sybase id string:
'Sybase  'Reserved:'       '
Backup Server: 4.38.1.1: Device '/dev/nrst0':
Label id:'HDR2'
Record format:'F'
Max. bytes/block:55296
Record length:02048
Backup format version:01
Reserved:'   '
Database name:'model                         '
Buffer offset length:00
Reserved:'                           '
```

After listing all files on a volume, the Backup Server sends a volume change request:

```
Backup Server: 6.30.1.2: Device /dev/nrst0: Volume cataloguing
complete.
Backup Server: 6.51.1.1: OPERATOR: Mount the next volume to search.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
@session_id = 5,
          @devname = '/dev/nrst0',
          @action = { 'PROCEED' | 'RETRY' | 'ABORT' },
          @fname = '
```

The operator can use sp_volchanged to mount another volume and signal the volume change or to terminate the search operation for all stripe devices.

Copying the log after a device failure

Normally, dump transaction truncates the inactive portion of the log after copying it. Use with no_truncate to copy the log without truncating it.

no_truncate allows you to copy the transaction log after failure of the device that holds your data. It uses pointers in the sysdatabases and sysindexes tables to determine the physical location of the transaction log. It can be used only if your transaction log is on a separate segment and your master database is accessible.

Warning! Use no_truncate only if media failure makes your data segment inaccessible. Never use no_truncate on a database that is in use.

Copying the log with no_truncate is the first step described in “Recovering a database: step-by-step instructions” on page 931.

Table 28-19 shows the syntax for copying a log after a device failure.

Table 28-19: Copying the log file after a device failure

Copying with the no_truncate option	

	`dump transaction` *`database_name`*
	 `to [compress::[`*`compression_level`*`::]]`
	 `stripe_device`
	 `[at` *`server_name`*`]`
	 `[density =` *`density,`*
	 `blocksize =` *`number_bytes,`*
	 `capacity =` *`number_kilobytes,`*
	 `dumpvolume =` *`volume_name`*`,`
	 `file =` *`file_name`*`]`
	 `[stripe on [compress::[`*`compression_level`*`::]]`
	 `stripe_device`
	 `[at` *`server_name`*`]`
	 `[density =` *`density`*`,`
	 `blocksize =` *`number_bytes`*`,`
	 `capacity =` *`number_kilobytes`*`,`
	 `dump volume =` *`volume_name`*`,`
	 `file =` *`file_name`*`] ...]`
	 `[with{`
	 `density =` *`density`*`,`
	 `blocksize =` *`number_bytes`*`,`
	 `capacity =` *`number_kilobytes`*`,`
	 `dumpvolume =` *`volume_name`*`,`
	 `file =` *`file_name`*`,`
	 `[nodismount | dismount],`
	 `[nounload | unload],`
	 `retaindays =` *`number_days`*`,`
	 `[noinit | init],`

Do not truncate log	`no_truncate,`

	`[notify = {client | operator_console}]`
	 `standby_access}]`

You can use no_truncate with striped dumps, tape initialization, and remote Backup Servers. Here is an example:

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

Truncating a log that is not on a separate segment

If a database does not have a log segment on a separate device from data segments, you cannot use dump transaction to copy the log and then truncate it. For these databases, you must:

1. Use the special with truncate_only option of dump transaction to truncate the log so that it does not run out of space
2. Use dump database to copy the entire database, including the log

Because it doesn't copy any data, with truncate_only requires only the name of the database:

```
dump transaction database_name with truncate_only
```

The following example dumps the database mydb, which does not have a log segment on a separate device from data segments, and then truncates the log:

```
dump database mydb to mydevice
dump transaction mydb with truncate_only
```

Truncating the log in early development environments

In early development environments, the transaction log is quickly filled by creating, dropping, and re-creating stored procedures and triggers and checking integrity constraints. Recovery of data may be less important than ensuring that there is adequate space on database devices.

with truncate_only allows you to truncate the transaction log without making a backup copy:

```
dump transaction database_name with truncate_only
```

After you run dump transaction with truncate_only, you must dump the database before you can run a routine log dump.

Truncating a log that has no free space

When the transaction log is very full, you may not be able to use your usual method to dump it. If you used dump transaction or dump transaction with truncate_only, and the command failed because of insufficient log space, use the special with no_log option of dump transaction:

```
dump transaction database_name with no_log
```

This option truncates the log without logging the dump transaction event. Because it doesn't copy any data, it requires only the name of the database.

Warning! Use dump transaction with no_log as a last resort, and use it only once after dump transaction with truncate_only fails. If you continue to load data after entering dump transaction with no_log, you may fill the log completely, causing any further dump transaction commands to fail. Use alter database to allocate additional space to the database.

All occurrences of dump tran with no_log are reported in the Adaptive Server error log. The message includes the user ID of the user executing the command. Messages indicating success or failure are also sent to the error log. no_log is the only dump option that generates error log messages.

Dangers of using *with truncate_only* and *with no_log*

with truncate_only and with no_log allow you to truncate a log that has become disastrously short of free space. Neither option provides a means to recover transactions that have committed since the last routine dump.

Warning! Run dump database at the earliest opportunity to ensure that your data can be recovered.

The following example truncates the transaction log for mydb and then dumps the database:

```
dump transaction mydb
    with no_log
dump database mydb to ...
```

Providing enough log space

Every use of dump transaction...with no_log is considered an error and is recorded in the server's error log. If you have created your databases with log segments on a separate device from data segments, written a last-chance threshold procedure that dumps your transaction log often enough, and allocated enough space to your log and database, you should not have to use this option.

However, some situations can still cause the transaction log to become too full, even with frequent log dumps. The dump transaction command truncates the log by removing all pages from the beginning of the log, up to the page preceding the page that contains an uncommitted transaction record (known as the oldest active transaction). The longer this active transaction remains uncommitted, the less space is available in the transaction log, since dump transaction cannot truncate additional pages.

This can happen when applications with very long transactions modify tables in a database with a small transaction log, which indicates you should increase the size of the log. It also occurs when transactions inadvertently remain uncommitted for long periods of time, such as when an implicit begin transaction uses the chained transaction mode or when a user forgets to complete the transaction. You can determine the oldest active transaction in each database by querying the syslogshold system table.

The *syslogshold* table

The syslogshold table is in the master database. Each row in the table represents either:

- The oldest active transaction in a database, or
- The Replication Server truncation point for the database's log.

A database may have no rows in syslogshold, a row representing one of the above, or two rows representing both of the above. For information about how a Replication Sever truncation point affects the truncation of the database's transaction log, see the Replication Server documentation.

Querying syslogshold provides a “snapshot” of the current situation in each database. Since most transactions last for only a short time, the query’s results may not be consistent. For example, the oldest active transaction described in the first row of syslogshold may finish before Adaptive Server completes the query of syslogshold. However, when several queries of syslogshold over time query the same row for a database, that transaction may prevent a dump transaction from truncating any log space.

When the transaction log reaches the last-chance threshold, and dump transaction cannot free up space in the log, you can query syslogshold and sysindexes to identify the transaction holding up the truncation. For example:

```
select H.spid, H.name
from master..syslogshold H, threshdb..sysindexes I
where H.dbid = db_id("threshdb")
and I.id = 8
and H.page = I.first

spid    name
------  -------------------------------------
     8  $user_transaction

(1 row affected)
```

This query uses the object ID associated with syslogs (8) in the threshdb database to match the first page of its transaction log with the first page of the oldest active transaction in syslogshold.

You can also query syslogshold and sysprocesses in the master database to identify the specific host and application owning the oldest active transactions. For example:

```
select P.hostname, P.hostprocess, P.program_name,
   H.name, H.starttime
from sysprocesses P, syslogshold H
where P.spid = H.spid
and H.spid != 0
```

```
hostname hostprocess program_name name              starttime
-------- ----------- ------------ ----------------- ------------------
eagle          15826 isql         $user_transaction  Sep  6 1997 4:29PM
hawk           15859 isql         $user_transaction  Sep  6 1997 5:00PM
condor         15866 isql         $user_transaction  Sep  6 1997 5:08PM

(3 rows affected)
```

Using the above information, you can notify or kill the user process owning the oldest active transaction and proceed with the dump transaction. You can also include the above types of queries in the threshold procedures for the database as an automatic alert mechanism. For example, you may decide that the transaction log should never reach its last-chance threshold. If it does, your last-chance threshold procedure (sp_thresholdaction) alerts you with information about the oldest active transaction preventing the transaction dump.

Note The initial log records for a transaction may reside in a user log cache, which is not visible in syslogshold until the records are flushed to the log (for example, after a checkpoint).

For more information about the syslogshold system table, see the *Reference Manual*. For information about the last-chance threshold and threshold procedures, see Chapter 31, “Managing Free Space with Thresholds.”

Responding to volume change requests

On UNIX and PC systems, use sp_volchanged to notify the Backup Server when the correct volumes have been mounted.

To use sp_volchanged, log in to any Adaptive Server that can communicate with both the Backup Server that issued the volume change request and the Adaptive Server that initiated the dump or load.

sp_volchanged syntax

Use this syntax for sp_volchanged:

```
sp_volchanged session_id, devname, action
     [ ,fname  [, vname ] ]
```

- Use the *session_id* and *devname* parameters specified in the volume change request.
- *action* specifies whether to abort, proceed with, or retry the dump or load.

- *fname* specifies the file to load. If you do not specify a file name, Backup Server loads the file = *file_name* parameter of the load command. If neither sp_volchanged nor the load command specifies which file to load, the Backup Server loads the first file on the tape.
- The Backup Server writes the *vname* in the ANSI tape label when overwriting an existing dump, dumping to a brand new tape, or dumping to a tape whose contents are not recognizable. During loads, the Backup Server uses the *vname* to confirm that the correct tape has been mounted. If you do not specify a *vname*, the Backup Server uses the volume name specified in the dump or load command. If neither sp_volchanged nor the command specifies a volume name, the Backup Server does not check this field in the ANSI tape label.

Volume change prompts for dumps

This section describes the volume change prompts that appear while you are dumping a database or transaction log. Each prompt includes the possible operator actions and the appropriate sp_volchanged response.

- `Mount the next volume to search.`

  When appending a dump to an existing volume, the Backup Server issues this message if it cannot find the end-of-file mark.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, abort`
Mount a new volume and proceed with the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, proceed [,` *`fname`* `[,` *`vname`*`]]`

- `Mount the next volume to write.`

  The Backup Server issues this message when it reaches the end of the tape. This occurs when it detects the end-of-tape mark, dumps the number of kilobytes specified by the capacity parameter of the dump command, or dumps the *high* value specified for the device in the sysdevices system table.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, abort`

The operator can	By replying
Mount the next volume and proceed with the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, proceed[,` *`fname`* `[,` *`vname`*`]]`

- `Volume on device devname has restricted access (code access_code).`

  Dumps that specify the init option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a tape with ANSI access restrictions without specifying the init option.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, abort`
Mount another volume and retry the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, retry [,` *`fname`* `[,` *`vname`*`]]`
Proceed with the dump, overwriting any existing contents	`sp_volchanged` *`session_id`*`,` *`devname`*`, proceed[,` *`fname`* `[,` *`vname`*`]]`

- `Volume on device devname is expired and will be overwritten.`

  Dumps that specify the init option overwrite any existing contents of the tape. During dumps to single-file media, Backup Server issues this message if you have not specified the init option and the tape contains a dump whose expiration date has passed.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`devname`*`, abort`
Mount another volume and retry the dump	`sp_volchanged` *`session_id`*`,` *`session_id`*`, retry [,` *`session_id`* `[,` *`session_id`*`]]`
Proceed with the dump, overwriting any existing contents	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed[,` *`session_id`* `[,` *`session_id`*`]]`

- `Volume to be overwritten on 'devname' has not expired: creation date on this volume is creation_date, expiration date is expiration_date.`

  On single-file media, the Backup Server checks the expiration date of any existing dump unless you specify the init option. The Backup Server issues this message if the dump has not yet expired.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`session_id`*`, abort`
Mount another volume and retry the dump	`sp_volchanged` *`session_id`*`,` *`session_id`*`, retry [,` *`session_id`* `[,` *`session_id`*`]]`
Proceed with the dump, overwriting any existing contents	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed [,` *`session_id`* `[,` *`session_id`*`]]`

- `Volume to be overwritten on ’devname’ has unrecognized label data.`

  Dumps that specify the init option overwrite any existing contents of the tape. Backup Server issues this message if you try to dump to a new tape or a tape with non-Sybase data without specifying the init option.

The operator can	By replying
Abort the dump	`sp_volchanged` *`session_id`*`,` *`session_id`*`, abort`
Mount another volume and retry the dump	`sp_volchanged` *`session_id`*`,` *`session_id`*`, retry [,` *`session_id`* `[,` *`session_id`*`]]`
Proceed with the dump, overwriting any existing contents	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed [,` *`session_id`* `[,` *`session_id`*`]]`

Volume change prompts for loads

Following are the volume change prompts and possible operator actions during loads:

- `Dumpfile ’fname’ section vname found instead of ’fname’ section vname.`

  The Backup Server issues this message if it cannot find the specified file on a single-file medium.

The operator can	By replying
Abort the load	`sp_volchanged` *`session_id`*`,` *`session_id`*`, abort`

The operator can	By replying
Mount another volume and try to load it	`sp_volchanged` *`session_id`*`,` *`session_id`*`, retry [,` *`session_id`* `[,` *`session_id`*`]]`
Load the file on the currently mounted volume, even though it is not the specified file (not recommended)	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed [,` *`session_id`* `[,` *`session_id`*`]]`

- `Mount the next volume to read.`

  The Backup Server issues this message when it is ready to read the next section of the dump file from a multivolume dump.

The operator can	By replying
Abort the load	`sp_volchanged` *`session_id`*`,` *`session_id`*`, abort`
Mount the next volume and proceed with the load	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed [,` *`session_id`* `[,` *`session_id`*`]]`

- `Mount the next volume to search.`

  The Backup Server issues this message if it cannot find the specified file on multifile medium.

The operator can	By replying
Abort the load	`sp_volchanged` *`session_id`*`,` *`session_id`*`, abort`
Mount another volume and proceed with the load	`sp_volchanged` *`session_id`*`,` *`session_id`*`, proceed [,` *`session_id`* `[,` *`session_id`*`]]`

Recovering a database: step-by-step instructions

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption occurs, unless you are running dbcc commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. Adaptive Server marks the database as suspect and displays a warning message. If the disk storing the master database fails, users will not be able to log in to the server, and users already logged in will not be able to perform any actions that access the system tables in master.

When your database device fails, Sybase recommends the following steps:

1. Get a current log dump of *every database on the device*.
2. Examine the space usage of *every database on the device*.
3. After you have gathered this information for all databases on the device, drop each database.
4. Drop the failed device.
5. Initialize new devices.
6. Re-create the databases, one at a time.
7. Load the most recent database dump into each database.
8. Apply each transaction log dump in the order in which it was created.

These steps are described in detail in the following sections.

Getting a current dump of the transaction log

Use dump transaction with no_truncate to get a current transaction log dump *for each database on the failed device*. For example, to get a current transaction log dump of mydb:

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

Examining the space usage

The following steps are recommended to determine which devices your database uses, how much space is allocated on each device, and whether the space is used for data, log, or both. You can use this information when re-creating your databases to ensure that the log, data, and indexes reside on separate devices, and to preserve the scope of any user segments you have created.

Note You can also use these steps to preserve segment mappings when moving a database dump from one server to another (on the same hardware and software platform).

If you do not use this information to re-create the device allocations for damaged databases, Adaptive Server will *remap* the sysusages table after load database to account for discrepancies. This means that the database's system-defined and user-defined segments no longer match the appropriate device allocations. Incorrect information in sysusages can result in the log being stored on the same devices as the data, even if the data and the log were separate before recovery. It can also change user-defined segments in unpredictable ways, and can result in a database that cannot be created using a standard create database command.

To examine and record the device allocations for all damaged databases:

1 In master, examine the device allocations and uses for the damaged database:

```
select segmap, size from sysusages
    where dbid = db_id("database_name")
```

2 Examine the output of the query. Each row with a segmap of "3" represents a data allocation; each row with a segmap of "4" represents a log allocation. Higher values indicate user-defined segments; treat these as data allocations, to preserve the scope of these segments. The size column indicates the number of blocks of data. Note the order, use, and size of each disk piece.

For example, this output from a server that uses 2K logical pages translates into the sizes and uses described in Table 28-20:

```
segmap        size
-------       --------
     3          10240
     3           5120
     4           5120
```

```
                    8            1024
                    4            2048
```

Table 28-20: Sample device allocation

Device allocation	**Megabytes**
Data	20
Data	10
Log	10
Data (user-defined segment)	2
Log	4

Note If the segmap column contains 7s, your data and log are on the same device, and you can recover only up to the point of the most recent database dump. *Do not* use the log on option to create database. Just be sure that you allocate as much (or more) space than the total reported from sysusages.

3 Run sp_helpdb *database_name* for the database. This query lists the devices on which the data and logs are located:

```
name        db_size owner   dbid    created
-------     ------- ------  ----    -----------
mydb        46.0 MB sa        15    Apr  9 1991

status          device_fragments    size          usage
--------------  ----------------    -----         ------------
no options set  datadev1            20 MB         data only
                datadev2            10 MB         data only
                datadev3             2 MB         data only
                logdev1             10 MB         log only
                logdev1              4 MB         log only
```

Dropping the databases

After you have performed the preceding steps *for all databases on the failed device*, use drop database to drop each database.

Note If tables in other databases contain references to any tables in the database you are trying to drop, you must remove the referential integrity constraints with alter table before you can drop the database.

If the system reports errors because the database is damaged when you issue drop database, use the dropdb option of the dbcc dbrepair command:

```
dbcc dbrepair (mydb, dropdb)
```

See the *Error Message and Troubleshooting Guide* for more information about dbcc dbrepair.

Dropping the failed devices

After you have dropped each database, use sp_dropdevice to drop the failed device. See the *Reference Manual* for more information.

Initializing new devices

Use disk init to initialize the new database devices. See Chapter 16, “Initializing Database Devices,” for more information.

Re-creating the databases

Use the following steps to re-create each database using the segment information you collected earlier.

Note If you chose not to gather information about segment usage, use create database...for load to create a new database that is at least as large as the original.

---

1. Use create database with the for load option. Duplicate all device fragment mappings and sizes for each row of your database from the sysusages table, *up to and including the first log device*. Use the order of the rows as they appear in sysusages. (The results of sp_helpdb are in alphabetical order by device name, not in order of allocation.) For example, to re-create the mydb database allocations shown in Table 28-20 on page 933, enter:

   ```
   create database mydb
       on datadev1 = 20,
           datadev2 = 10
   log on logdev1 = 10
   for load
   ```

   Note create database...for load temporarily locks users out of the newly created database, and load database marks the database offline for general use. This prevents users from performing logged transactions during recovery.

2. Use alter database with the for load option to re-create the remaining entries, in order. Remember to treat device allocations for user segments as you would data allocations.

   In this example, to allocate more data space on datadev3 and more log space on logdev1, the command is:

   ```
   alter database mydb
       on datadev3 = "2M"
   log on logdev1= "4M"
   for load
   ```

Loading the database

Reload the database using load database. If the original database stored objects on user-defined segments (sysusages reports a segmap greater than 7) and your new device allocations match those of the dumped database, Adaptive Server preserves the user segment mappings.

If you did not create the new device allocations to match those of the dumped database, Adaptive Server will remap segments to the available device allocations. This remapping may also mix log and data on the same physical device.

Note If an additional failure occurs while a database is being loaded, Adaptive Server does not recover the partially loaded database, and notifies the user. You must restart the database load by repeating the load command.

Loading the transaction logs

Use load transaction to apply transaction log backups *in the same sequence in which they were made*.

Adaptive Server checks the timestamps on each dumped database and transaction log. If the dumps are loaded in the wrong order, or if user transactions have modified the transaction log between loads, the load fails.

If you dumped the transaction log using with standby_access, you must also load the database using standby_access.

After you have brought a database up to date, use dbcc commands to check its consistency.

Loading a transaction log to a point in time

You can recover a database up to a specified point in time in its transaction log. To do so, use the until_time option of load transaction. This is useful if, for example, a user inadvertently drops an important table; you can use until_time to recover the changes made to the database containing the table up to a time just before the table was dropped.

To use until_time effectively after data has been destroyed, you must know the exact time the error occurred. You can find this by issuing a select getdate at the time of the error. For example, suppose a user accidentally drops an important table, and then a few minutes later you get the current time in milliseconds:

```
select convert(char(26), getdate(), 109)

--------------------------
```

```
Mar 26 1997 12:45:59:650PM
```

After dumping the transaction log containing the error and loading the most recent database dump, load the transaction logs that were created after the database was last dumped. Then, load the transaction log containing the error by using until_time; for example:

```
load transaction employees_db
from "/dev/nrmt5"
with until_time = "Mar 26 1997 12:35:59: 650PM"
```

After you load a transaction log using until_time, Adaptive Server restarts the database's log sequence. This means that until you dump the database again, you cannot load subsequent transaction logs after the load transaction using until_time. You will need to dump the database before you can dump another transaction log.

Bringing the databases online

In this example, the transaction log is loaded up to a time just before the table drop occurred. After you have applied all transaction log dumps to a database, use online database to make it available for use. In this example, the command to bring the mydb database online is:

```
online database mydb
```

Replicated databases

Before you upgrade replicated databases to the current version of Adaptive Server, the databases must be online. However, you cannot bring replicated databases online until the logs are drained. If you try to bring a replicated database online before the logs are drained, Adaptive Server issues the following message:

```
Database is replicated, but the log is not yet
drained. This database will come online
automatically after the log is drained.
```

When Replication Server, via the Log Transfer Manager (LTM), drains the log, online database is automatically issued.

Upgrading to the current release of Adaptive Server

Refer to the installation documentation for your platform for upgrade instructions for Adaptive Server users that have replicated databases.

Load sequence

The load sequence for loading replicated databases is: load database, replicate, load transaction, replicate, and so on. At the end of the load sequence, issue online database to bring the databases online. Databases that are offline because they are in a load sequence are not automatically brought online by Replication Server.

Warning! Do not issue online database until all transaction logs are loaded.

Loading database dumps from older versions

When you upgrade an Adaptive Server installation is upgraded to a new version, all databases associated with that server are automatically upgraded.

As a result, database and transaction log dumps created with a previous version of Adaptive Server must be upgraded before they can be used with the current version of Adaptive Server.

Adaptive Server provides an automatic upgrade mechanism – on a per-database basis – for upgrading a database or transaction log made with Backup Sever to the current Adaptive Server version, thus making the dump compatible for use. This mechanism is entirely internal to Adaptive Server, and requires no external programs. It provides the flexibility of upgrading individual dumps as needed.

The following tasks are not supported by this automatic upgrade functionality:

- Loading an older release of the master database. That is, if you upgraded Adaptive Server to the current version, you cannot load a dump of the master database from which you upgraded.
- Installing new or modified stored procedures. Continue to use installmaster.
- Loading and upgrading dumps generated previous to SQL Server release 10.0.

How to upgrade a dump to Adaptive Server

To upgrade a user database or transaction log dump to the current version of Adaptive Server:

1. Use load database and load transaction to load the dump to be upgraded.

   Adaptive Server determines from the dump header which version it is loading. After the dump header is read, and before Backup Server begins the load, the database is marked offline by load database or load transaction. This makes the database unavailable for general use (queries and use *database* are not permitted), provides the user greater control over load sequences, and eliminates the possibility that other users will accidentally interrupt a load sequence.

2. Use online database, after the dump has successfully loaded, to activate the upgrade process.

   Note Do *not* issue online database until after all transaction dumps are loaded.

   Prior to SQL Server version 11.0, a database was automatically available at the end of a successful load sequence. With the current version of Adaptive Server, the user is required to bring the database online after a successful load sequence, using online database.

   For dumps loaded from SQL Server version 10.0, online database activates the upgrade process to upgrade the dumps just loaded. After the upgrade is successfully completed, Adaptive Server places the database online and the database is ready for use.

   For dumps loaded from the current version of Adaptive Server, no upgrade process is activated. You must still issue online database to place the database online – load database marks it as offline.)

   Each upgrade step produces a message stating what it is about to do.

   An upgrade failure leaves the database offline and produces a message stating that the upgrade failed and the user must correct the failure.

   For more information about online database, see the *Reference Manual*.

3 After successful execution of online database, use dump database. The database must be dumped before a dump transaction is permitted. A dump transaction on a newly created or upgraded database is not permitted until a successful dump database has occurred.

The *database offline* status bit

The “database offline” status bit indicates that the database is not available for general use. You can determine whether a database is offline by using sp_helpdb. It will show that the database is offline if this bit is set.

When a database is marked offline by load database, a status bit in the sysdatabases table is set and remains set until the successful completion of online database.

The “database offline” status bit works in combination with any existing status bits. It augments the following status bit to provide additional control:

- In recovery

The “database offline” status bit overrides the following status bits:

- DBO use only
- Read only

The following status bits override the “database offline” status bit:

- Began upgrade
- Bypass recovery
- In load
- Not recovered
- Suspect
- Use not recovered

Although the database is not available for general use, you can user these commands when the database is offline:

- dump database and dump transaction
- load database and load transaction
- alter database on device

- drop database
- online database
- dbcc diagnostics (subject to dbcc restrictions)

Version identifiers

The automatic upgrade feature provides version identifiers for Adaptive Server, databases, and log record formats:

- Configuration upgrade version ID – shows the current version of Adaptive Server; it is stored in the sysconfigures table. sp_configure displays the current version of Adaptive Server as “upgrade version.”
- Upgrade version indicator – shows the current version of a database and is stored in the database and dump headers. The Adaptive Server recovery mechanism uses this value to determine whether the database should be upgraded before being made available for general use.
- Log compatibility version specifier – differentiates version 10.x logs from version 11.x logs by showing the format of log records in a database, database dump, or transaction log dump. This constant is stored in the database and dump headers and is used by Adaptive Server to detect the format of log records during recovery.

Cache bindings and loading databases

If you dump a database and load it onto a server with different cache bindings, you should be aware of cache bindings for a database and the objects in the database. You may want to load the database onto a different server for tuning or development work, or you may need to load a database that you dropped from a server whose cache bindings have changed since you made the dump.

When you bring a database online after recovery or by using online database after a load, Adaptive Server verifies all cache bindings for the database and database objects. If a cache does not exist, Adaptive Server writes a warning to the error log, and the binding in sysattributes is marked as invalid. Here is an example of the message from the error log:

```
Cache binding for database '5', object '208003772',
index '3' is being marked invalid in Sysattributes.
```

Invalid cache bindings are not deleted. If you create a cache of the same name and restart Adaptive Server, the binding is marked as valid and the cache is used. If you do not create a cache with the same name, you can bind the object to another cache or allow it to use the default cache.

In the following sections, which discuss cache binding topics, *destination server* refers to the server where the database is being loaded, and *original server* refers to the server where the dump was made.

If possible, re-create caches that have the same names on the destination server as the bindings on the original server. You may want to configure pools in exactly the same manner if you are using the destination database for similar purposes or for performance testing and development that may be ported back to the original server. If you are using the destination database for decision support or for running dbcc commands, you may want to configure pools to allow more space in 16K memory pools.

Databases and cache bindings

Binding information for databases is stored in master..sysattributes. No information about database binding is stored in the database itself. If you use load database to load the dump over an existing database that is bound to a cache, and you do not drop the database before you issue the load command, this does not affect the binding.

If the database that you are loading was bound to a cache on the original server, you can:

- Bind the database on the destination server to a cache configured for the needs on that server, or
- Configure pools in the default data cache on the destination server for the needs of the application there, and do not bind the database to a named data cache.

Database objects and cache bindings

Binding information for objects is stored in the sysattributes table in the database itself. If you frequently load the database onto the destination server, the simplest solution is to configure caches of the same name on the destination server.

If the destination server is not configured with caches of the same name as the original server, bind the objects to the appropriate caches on the destination server after you bring the database online, or be sure that the default cache is configured for your needs on that server.

Checking on cache bindings

Use sp_helpcache to display the cache bindings for database objects, even if the cache bindings are invalid.

The following SQL statements reproduce cache binding commands from the information in a user database's sysattributes table:

```
/* create a bindcache statement for tables */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", " + object_name(object)
from sysattributes
where class = 3
    and object_type = "T"

/* create a bindcache statement for indexes */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", "   + i.name
from sysattributes, sysindexes i
where class = 3
    and object_type = "I"
    and i.indid = convert(tinyint, object_info1)
    and i.id = object
```

Cross-database constraints and loading databases

If you use the references constraint of create table or alter database to reference tables across databases, you may encounter problems when you try to load a dump of one of these databases.

- If tables in a database reference a dumped database, referential integrity errors result if you load the database with a different name or on a different server from where it was dumped. To change the name or location of a database when you reload it, use alter database in the referencing database to drop all external referential integrity restraints before you dump the database.
- Loading a dump of a referenced database that is earlier than the referencing database could cause consistency issues or data corruption. As a precaution, each time you add or remove a cross-database constraint or drop a table that contains a cross-database constraint, dump both affected databases.
- Dump all databases that reference each other at the same time. To guard against synchronization problems, put both databases in single-user mode for the dumps. When loading the databases, bring both databases online at the same time.

Cross-database constraints can become inconsistent if you:

- Do not load database dumps in chronological order (for example, you load a dump created on August 12, 1997, after one created on August 13), or
- Load a dump into a database with a new name.

If you do not load, cross-database constraints can become inconsistent.

To remedy this problem:

1. Put both databases in single-user mode.
2. Drop the inconsistent referential constraint.
3. Check the data consistency with a query such as:

```
select foreign_key_col from table1
where foreign_key not in
(select primary_key_col from otherdb..othertable)
```

4. Fix any data inconsistency problems.
5. Re-create the constraint.

CHAPTER 29 Restoring the System Databases

This chapter explains how to restore the master, model, and sybsystemprocs databases.

What does recovering a system database entail?

The recovery procedure for system databases depends on the database involved and the problems that you have on your system. In general, recovery may include:

- Using load database to load backups of these databases,
- Using dataserver, installmaster, and installmodel to restore the initial state of these databases, or
- A combination of the above tasks.

To make the recovery of system databases as efficient as possible:

- Do not store user databases or any databases other than master, tempdb, and model on the master device.
- Always keep up-to-date printouts of important system tables.
- Always back up the master database after performing actions such as initializing database devices, creating or altering databases, or adding new server logins.

Symptoms of a damaged *master* database

A damaged master database can be caused by a media failure in the area on which master is stored or by internal corruption in the database. You'll know if your master database is damaged if:

- Adaptive Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- dbcc reports damage during a regular check of your databases.

Recovering the *master* database

This section describes how to recover the master database and to rebuild the master device. It assumes:

- The master database is corrupt, or the master device is damaged.
- You have up-to-date printouts of the system tables, listed in "Backing up master and keeping copies of system tables" on page 24.
- The master device contains *only* the master database, tempdb, and model.
- You have an up-to-date backup of the master database, and you have not initialized any devices or created or altered any databases since last dumping master.
- Your server uses the default sort order.

You can also use these procedures to move your master database to a larger master device.

The *Error Message and Troubleshooting Guide* provides more complete coverage of recovery scenarios.

About the recovery process

Special procedures are needed because of the central, controlling nature of the master database and the master device. Tables in master configure and control all Adaptive Server's functions, databases, and data devices. The recovery process:

- Rebuilds the master device to its default state when you first installed a server
- Restores the master database to the default state
- Restores the master database to its condition at the time of your last backup

During the early stages of recovering the master database, you cannot use the system stored procedures.

Summary of recovery procedure

You must follow the steps below to restore a damaged master device. *Each step is discussed in more detail on the following pages.*

Step	See
Find hard copies of the system tables needed to restore disks, databases and logins.	"Step 1: Find copies of system tables" on page 950
Shut down Adaptive Server, and use dataserver to build a new master database and master device.	"Step 2: Build a new master device" on page 950
Restart Adaptive Server in master-recover mode.	"Step 3: Start Adaptive Server in master-recover mode" on page 951
Re-create the master database's allocations in sysusages exactly.	"Step 4: Re-create device allocations for master" on page 952
Update Backup Server's network name in the sysservers table.	"Step 5: Check your Backup Server sysservers information" on page 957
Verify that your Backup Server is running.	"Step 6: Verify that your Backup Server is running" on page 958
Use load database to load the most recent database dump of master. Adaptive Server stops automatically after successfully loading master.	"Step 7: Load a backup of master" on page 958
Update the number of devices configuration parameter in the configuration file.	"Step 8: Update the number of devices configuration parameter" on page 958.
Restart Adaptive Server in single-user mode.	"Step 9: Restart Adaptive Server in master-recover mode" on page 959

Step	See
Verify that the backup of master has the latest system tables information.	"Step 10: Check system tables to verify current backup of master" on page 959
Restart Adaptive Server.	"Step 11: Restart Adaptive Server" on page 960
Check syslogins if you have added new logins since the last backup of master.	"Step 12: Restore server user IDs" on page 960
Restore the model database.	"Step 13: Restore the model database" on page 961
Compare hard copies of sysusages and sysdatabases with the new online version, run dbcc checkalloc on each database, and examine the important tables in each database.	"Step 14: Check Adaptive Server" on page 961
Dump the master database.	"Step 15: Back up master" on page 962

Step 1: Find copies of system tables

Find copies of the system tables that you have saved to a file: sysdatabases, sysdevices, sysusages, sysloginroles, and syslogins. You can use these to guarantee that your system has been fully restored at the completion of this process.

For information on preparing for disaster recovery by making copies of the system tables to a file, see "Backing up master and keeping copies of system tables" on page 24.

Step 2: Build a new master device

Before you run dataserver, check your most recent copy of sysusages. If it has only one line for dbid 1, your master database has only one disk allocation piece, and you can go to "Step 5: Check your Backup Server sysservers information" on page 957.

Shut down Adaptive Server, if it is running, and rebuild the master device. When rebuilding the master device, you must specify the device size.

Before you begin, it is important that you remember to:

- Use a new device, preserving the old device in case you encounter problems. The old device may provide crucial information.
- Shut down Adaptive Server before you use any dataserver command. If you use dataserver on a master device that is in use by Adaptive Server, the recovery procedure will fail when you attempt to load the most recent backup of master.

Run dataserver (UNIX) or sqlsrvr (Windows NT) to build a new master device and to install a copy of a "generic" master database. Give the full name and full size for your master device.

Note You must give dataserver a size as large as or larger than the size originally used to configure Adaptive Server. If the size is too small, you will get error messages when you try to load your databases.

The following example rebuilds a 17MB master device.

- On UNIX platforms:

```
dataserver -d /dev/rsd1f -b17M
```

- On Windows NT:

```
sqlsrvr -d d:\devices\master.dat -b17M
```

After you run dataserver, the password for the default "sa" account reverts to NULL.

For details on the dataserver utility, see the *Utility Guide*.

Step 3: Start Adaptive Server in master-recover mode

Start Adaptive Server in master-recover mode with the -m (UNIX and Windows NT) option.

- On UNIX platforms – make a copy of the runserver file, naming it *m_RUN_server_name*. Edit the new file, adding the parameter -m to the dataserver command line. Then start the server in master-recover mode:

```
startserver -f m_RUN_server_name
```

- On Windows NT – start Adaptive Server from the command line using the sqlsrver command. Specify the -m parameter in addition to other necessary parameters. For example:

```
sqlsrver.exe -dD:\Sybase\DATA\MASTER.dat -sPIANO
-eD:\Sybase\install\errorlog -iD:\Sybase\ini -MD:\Sybase -m
```

See the *Utility Guide* for the complete syntax of these commands.

When you start Adaptive Server in master-recover mode, only one login of one user—the System Administrator—is allowed. Immediately following a dataserver command on the master database, only the “sa” account exists, and its password is NULL.

Warning! Some sites have automatic jobs that log in to the server at start-up with the “sa” login. Be sure these are disabled.

Master-recover mode is necessary because the generic master database created with dataserver does not match the actual situation in Adaptive Server. For example, the database does not know about any of your database devices. Any operations on the master database could make recovery impossible or at least much more complicated and time-consuming.

An Adaptive Server started in master-recover mode is automatically configured to allow direct updates to the system tables. Certain other operations (for example, the

process) are disallowed.

Warning! Ad hoc changes to system tables are dangerous—some changes can render Adaptive Server unable to run. Make only the changes described in this chapter, and always make the changes in a user-defined transaction.

Step 4: Re-create device allocations for *master*

If more than one row for dbid 1 appears in your hard copy of sysusages, you need to increase the size of master so that you can load the dump. You must duplicate the vstart value for each allocation for master in sysusages. This is easiest to do if you have a copy of sysusages ordered by vstart.

In the simplest cases, additional allocations to master require only the use of alter database. In more complicated situations, you must allocate space for other databases to reconstruct the exact vstart values needed to recover master.

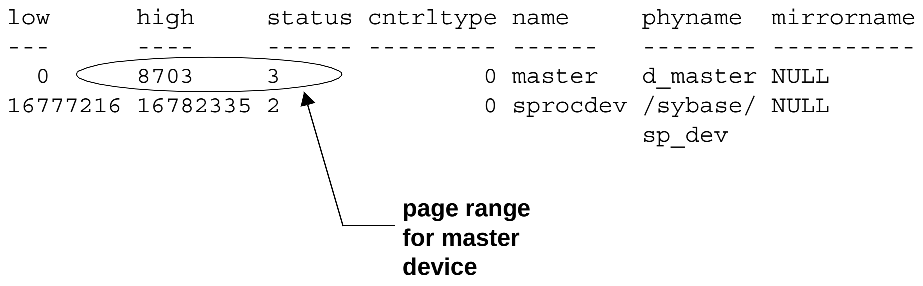
In addition to the master database, tempdb (dbid = 2) and model (dbid = 3) are located wholly or partially on the master device.

Determining which allocations are on the master device

Note The examples in this section assume a server that uses 2K logical pages.

To determine which vstart values represent allocations on the master device, look at the sysdevices table. It shows the low and high values for each device. Databases devices always have a cntrltype of 0; the following example does not include the rows for tape devices.

Figure 29-1: Determining allocations on the master device

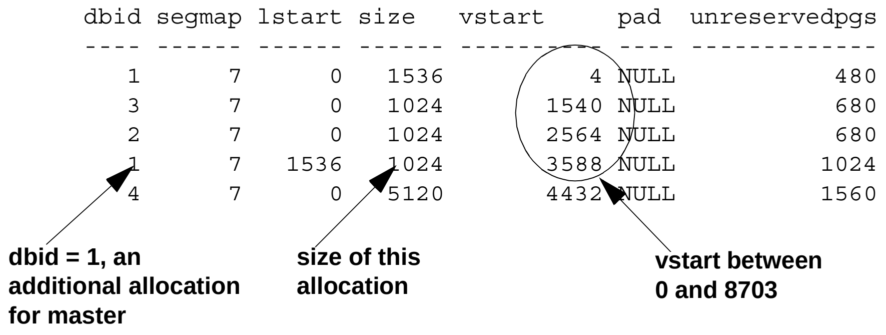


```
low      high     status cntrltype name     phyname  mirrorname
---      ----     ------ --------- ------   -------- ----------
  0        8703     3            0 master   d_master NULL
16777216 16782335 2              0 sprocdev /sybase/ NULL
                                            sp_dev
```

In this example, page numbers on the master device are between 0 and 8703, so any database allocation with sysusages.vstart values in this range represent allocations on the master device.

Check all rows for master (except the first) in your saved sysusages output. Here is sample sysusages information, ordered by vstart:

Figure 29-2: Sample output from sysusages



```
dbid segmap lstart size   vstart     pad  unreservedpgs
---- ------ ------ ------ ---------- ---- -------------
   1      7      0   1536          4 NULL           480
   3      7      0   1024       1540 NULL           680
   2      7      0   1024       2564 NULL           680
   1      7   1536   1024       3588 NULL          1024
   4      7      0   5120       4432 NULL          1560
```

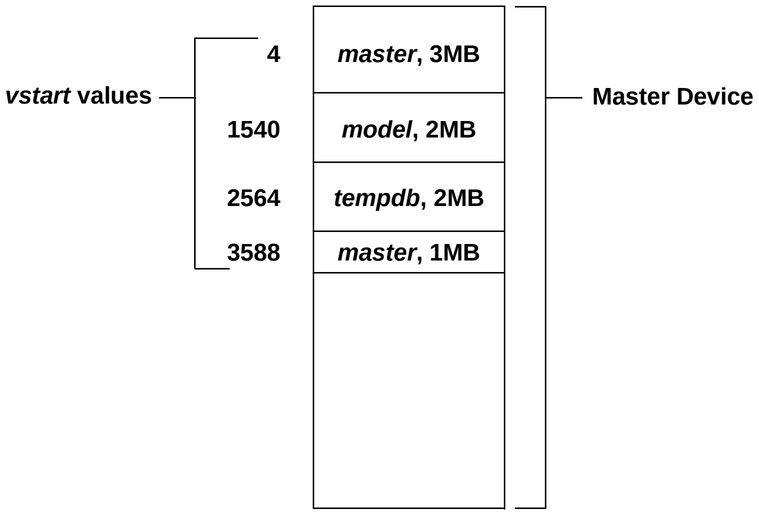
In this example, the first four rows have vstart values between 4 and 3588. Only dbid 4 is on another device.

dataserver re-creates the first three rows, so sysusages in your newly rebuilt master database should match your hard copy.

The fourth row shows an additional allocation for master with vstart = 3588 and size = 1024.

Figure 29-3 shows the storage allocations for the above sysusages data.

Figure 29-3: Allocations on a master device



In Figure 29-3, you need only to issue an alter database command to increase the size of the master database. To determine the size to provide for this command, look at the size column for the second allocation to master. Divide by 512. In this example, the additional row for master indicates an allocation of 1024 data pages, so the correct parameter is 2, the result of 1024/512.

Use that result for the alter database command. Log in to the server as “sa.” Remember that dataserver has set the password for this account to NULL. Issue the alter database command. For the example above, use:

```
alter database master on master = "2M"
```

Check the size and vstart values for the new row in sysusages.

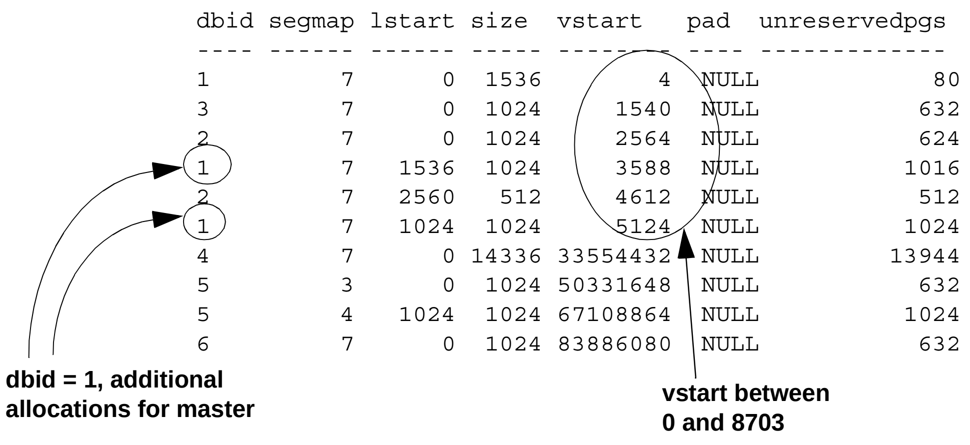
Creating additional allocations

Your output from sysusages may have more allocations on the master device if:

- You have upgraded Adaptive Server from an earlier version
- A System Administrator has increased the size of master, model, or tempdb on the master device

You must restore these allocations up to the last row for the master database, dbid 1. Here is an example of sysusages showing additional allocations on the master device, in vstart order:

Figure 29-4: Sample sysusages output with additional allocations



```
dbid segmap lstart size  vstart   pad  unreservedpgs
---- ------ ------ ----- -------- ---- -------------
1         7      0  1536        4  NULL            80
3         7      0  1024     1540  NULL           632
2         7      0  1024     2564  NULL           624
1         7   1536  1024     3588  NULL          1016
2         7   2560   512     4612  NULL           512
1         7   1024  1024     5124  NULL          1024
4         7      0 14336 33554432  NULL         13944
5         3      0  1024 50331648  NULL           632
5         4   1024  1024 67108864  NULL          1024
6         7      0  1024 83886080  NULL           632
```

dbid = 1, additional allocations for master
vstart between 0 and 8703

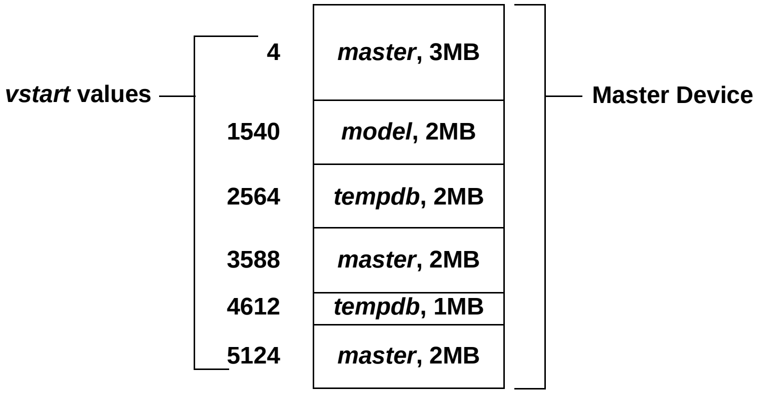
This copy of sysusages shows the following allocations on the master device (excluding the three created by dataserver):

- One for master, dbid = 1, size = 1024, vstart = 3588
- One for tempdb, dbid = 2, size = 512, vstart = 4612
- Another allocation for master, dbid = 1, size = 1024, vstart = 5124

The final allocations in this output are not on the master device.

Figure 29-5 shows the allocations on the master device.

Figure 29-5: Complex allocations on a master device



You need to issue a set of alter database and create database commands to re-create all the allocations with the correct sizes and vstart values. If sysusages lists additional allocations on the master device after the last allocation for master, you do not have to re-create them.

To determine the size for the create database and alter database commands, divide the value shown in the size column of the sysusages output by 512.

To reconstruct the allocation, issue these commands, in this order:

- To restore the first allocation to master, dbid 1, size = 1024:

```
alter database master on default = "2M"
```

- To allocate more space to tempdb, dbid 2, size = 512:

```
alter database tempdb on default = "1M"
```

- To add the final allocation to master, dbid 1, size = 1024:

```
alter database master on default = "1M"
```

You need to restore only the allocations up to and including the last line for the master database. When you load the backup of master, this table is completely restored from the dump.

At this point, carefully check the current sysusages values with the values in your hard copy:

- If all of the vstart and size values for master match, go to “Step 5: Check your Backup Server sysservers information” on page 957.

- If the values do not match, an attempt to load the master database will almost certainly fail. Shut down the server, and begin again by running dataserver. See “Step 2: Build a new master device” on page 950.
- If your sysusages values look correct, go to “Step 5: Check your Backup Server sysservers information” on page 957.

Step 5: Check your Backup Server *sysservers* information

Log in to the server as “sa,” using a null password.

If the network name of your Backup Server is not SYB_BACKUP, you must update sysservers so that Adaptive Server can communicate with its Backup Server. Check the Backup Server name in your interfaces file, and issue this command:

```
select *
from sysservers
where srvname = "SYB_BACKUP"
```

Check the srvnetname in the output from this command. If it matches the interfaces file entry for the Backup Server for your server, go to “Step 6: Verify that your Backup Server is running” on page 958.

If the reported srvnetname is *not* the same as the Backup Server in the interfaces file, you must update sysservers. The example below changes the Backup Server’s network name to PRODUCTION_BSRV:

```
begin transaction
update sysservers
set srvnetname = "PRODUCTION_BSRV"
where srvname = "SYB_BACKUP"
```

Execute this command, and check to be sure that it modified only one row. Issue the select command again, and verify that the correct row was modified and that it contains the correct value. If update modified more than one row, or if it modified the wrong row, issue a rollback transaction command, and attempt the update again.

If the command correctly modified the Backup Server’s row, issue a commit transaction command.

Step 6: Verify that your Backup Server is running

On UNIX platforms, use the showserver command to verify that your Backup Server is running; restart your Backup Server if necessary. See showserver and startserver in the *Utility Guide*.

On Windows NT, a locally installed Sybase Central and the Services Manager show whether Backup Server is running.

See the *Utility Guide* for the commands to start Backup Server.

Step 7: Load a backup of *master*

Load the most recent backup of the master database. Here are examples of the load commands:

- On UNIX platforms:

```
load database master from "/dev/nrmt4"
```

- On Windows NT:

```
load database master from "\\.\TAPE0"
```

See Chapter 28, “Backing Up and Restoring User Databases,” for information on command syntax.

After load database completes successfully, Adaptive Server shuts down. Watch for any error messages during the load and shut down processes.

Step 8: Update the *number of devices* configuration parameter

Perform this step only if you use more than the default number of database devices. Otherwise, go to “Step 9: Restart Adaptive Server in master-recover mode” on page 959.

Configuration values are not available to Adaptive Server until after recovery of the master database, so you need to instruct Adaptive Server to read the appropriate value for the number of devices parameter from a configuration file at start-up.

If your most recent configuration file is not available, edit a configuration file to reflect the correct value for the number of devices parameter.

Edit the runserver file. Add the -c parameter to the end of the dataserver or sqlsrver command, specifying the name and location of the configuration file. When Adaptive Server starts, it reads the parameter values from the specified configuration file.

Step 9: Restart Adaptive Server in master-recover mode

Use startserver to restart Adaptive Server in master-recover mode (see "Step 3: Start Adaptive Server in master-recover mode" on page 951). Watch for error messages during recovery.

Loading the backup of master restores the "sa" account to its previous state. It restores the password on the "sa" account, if one exists. If you used sp_locklogin to lock this account before the backup was made, the "sa" account will now be locked. Perform the rest of the recovery steps using an account with the System Administrator role.

Step 10: Check system tables to verify current backup of *master*

If you have backed up the master database since issuing the most recent disk init, create database, or alter database command, then the contents of sysusages, sysdatabases, and sysdevices will match your hard copy.

Check the sysusages, sysdatabases, and sysdevices tables in your recovered server against your hard copy. Look especially for these problems:

- If any devices in your hard copy are not included in the restored sysdevices, then you have added devices since your last backup, and you must run disk reinit and disk refit. For information on using these commands, see "Restoring system tables with disk reinit and disk refit" on page 966.

- If any databases listed in your hard copy are not listed in your restored sysdatabases table, you have added a database since the last time you backed up master. You must run disk refit (see "Restoring system tables with disk reinit and disk refit" on page 966).

Note You must start Adaptive Server with trace flag 3608 before you run disk refit. However, make sure you read the information provided in the *Troubleshooting and Error Messages Guide* before you start Adaptive Server with any trace flag.

Step 11: Restart Adaptive Server

Restart Adaptive Server in normal (multiuser) mode.

Step 12: Restore server user IDs

Check your hard copy of syslogins and your restored syslogins table. Look especially for the following situations and reissue the appropriate commands, as necessary:

- If you have added server logins since the last backup of master, reissue the sp_addlogin commands.
- If you have dropped server logins, reissue the sp_droplogin commands.
- If you have locked server accounts, reissue the sp_locklogin commands.
- Check for other differences caused by the use of sp_modifylogin by users or by System Administrators.

Make sure that the suids assigned to users are correct. Mismatched suid values in databases can lead to permission problems, and users may not be able to access tables or run commands.

An effective technique for checking existing suid values is to perform a union on each sysusers table in your user databases. You can include master in this procedure, if users have permission to use master.

For example:

```
select suid, name from master..sysusers
```

```
union
select suid, name from sales..sysusers
union
select suid, name from parts..sysusers
union
select suid, name from accounting..sysusers
```

If your resulting list shows skipped suid values in the range where you need to redo the logins, you must add placeholders for the skipped values and then drop them with sp_droplogin or lock them with sp_locklogin.

Step 13: Restore the *model* database

Restore the model database:

- Load your backup of model, if you keep a backup.
- If you do not have a backup:
  - Run the installmodel script:

    On most platforms:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < installmodel
```

    On Windows NT:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < instmodl
```

  - Redo any changes you made to model.

Step 14: Check Adaptive Server

Check Adaptive Server carefully:

1. Compare your hard copy of sysusages with the new online version.
2. Compare your hard copy of sysdatabases with the new online version.
3. Run dbcc checkalloc on each database.

4 Examine the important tables in each database.

Warning! If you find discrepancies in sysusages, call Sybase Technical Support.

Step 15: Back up *master*

When you have completely restored the master database and have run full dbcc integrity checks, back up the database using your usual dump commands.

Recovering the *model* database

This section describes recovery of the model database when only the model database needed to be restored. It includes instructions for these scenarios:

- You have not made any changes to model, so you need to restore only the generic model database.
- You have changed model, and you have a backup.
- You have changed model, and you do not have a backup.

Restoring the generic *model* database

dataserver can restore the model database without affecting master.

Warning! Shut down Adaptive Server before you use any dataserver command.

- On UNIX platforms:

```
dataserver -d /devname  -x
```

- On Windows NT:

```
sqlsrvr -d physicalname -x
```

Restoring *model* from a backup

If you can issue the model successfully, you can restore your model database from a backup with load database.

If you cannot use the database:

1. Follow the instructions for “Restoring the generic model database” on page 962.
2. If you have changed the size of model, reissue alter database.
3. Load the backup with load database.

Restoring *model* with no backup

If you have changed your model database, and you do not have a backup:

- Follow the steps for “Restoring the generic model database” on page 962.
- Reissue all the commands you issued to change model.

Recovering the *sybsystemprocs* database

The sybsystemprocs database stores the system procedures that are used to modify and report on system tables. If your routine dbcc checks report damage, and you do not keep a backup of this database, you can restore it using installmaster. If you do keep backups of sybsystemprocs, you can restore it with load database.

Restoring *sybsystemprocs* with *installmaster*

1. Check to see what logical device currently stores the database. If you can still use sp_helpdb, issue:

```
                          sp_helpdb sybsystemprocs
     name                 db_size       owner           dbid
             created
             status
     ------------------ ------------ --------------- ------
```

```
sybsystemprocs           28.0 MB sa                  4
        Aug 07, 1993
        trunc log on chkpt

device_fragments   size        usage            free kbytes
------------------ ----------- ---------------- -----------
sprocdev           28.0 MB     data and log            3120
```

The “device_fragments” column indicates that the database is stored on sprocdev.

If you cannot use sp_helpdb, this query reports the devices used by the database and the amount of space on each device:

```
select sysdevices.name, sysusages.size / 512
from sysdevices, sysdatabases, sysusages
where sysdatabases.name = "sybsystemprocs"
  and sysdatabases.dbid = sysusages.dbid
  and sysdevices.low <= sysusages.size + vstart
  and sysdevices.high >= sysusages.size + vstart -1
name
--------------- -------
sprocdev             28
```

2 Drop the database:

```
drop database sybsystemprocs
```

If the physical disk is damaged, use dbcc dbrepair to drop the database and then use sp_dropdevice to drop the device (sp_dropdevice is stored in master, so it still exists even though you dropped sybsystemprocs). If necessary, use disk init to initialize a new database device. See Chapter 16, “Initializing Database Devices,” for more information on disk init.

3 Re-create the sybsystemprocs database on the device, using the size returned by the query under step 1:

```
create database sybsystemprocs
```

```
    on sprocdev = 28
```

Note The required size for sybsystemprocs may be different for your operating system. See the installation documentation for your platform for the correct size.

4 Run the installmaster script.

Warning! Running *installmaster* repeatedly can change the distribution of index values in such a way that the sysprocedures table will require much more disk space to store the same amount of data. To avoid this problem, drop and re-create sybsystemprocs before running *installmaster*.

- On UNIX platforms:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name <
installmaster
```

- On Windows NT:

```
cd $SYBASE/scripts
isql -Usa -Ppassword -Sserver_name < instmstr
```

5 If you have made any changes to permissions in sybsystemprocs, or if you have added your own procedures to the database, you must redo the changes.

Restoring *sybsystemprocs* with *load database*

If you write system procedures and store them in sybsystemprocs, there are two ways to recover them if the database is damaged:

- Restore the database from installmaster, as described in step 4 under “Restoring sybsystemprocs with installmaster” on page 963. Then re-create the procedures by reissuing the create procedure commands.
- Keep backups of the database, and load them with load database.

If you choose to keep a backup of the database, be sure that the complete backup fits on one tape volume or that more than one Adaptive Server is able to communicate with your Backup Server. If a dump spans more than one tape volume, issue the change-of-volume command using sp_volchanged, which is stored in sybsystemprocs. You cannot issue that command in the middle of recovering a database.

Following are sample load commands:

- On UNIX:

```
load database sybsystemprocs from "/dev/nrmt4"
```

- On Windows NT:

```
load database sybsystemprocs from "\\.\TAPE0"
```

Restoring system tables with *disk reinit* and *disk refit*

When you are restoring the master database from a dump that does not reflect the most recent disk init or create database and alter database commands, follow the procedures in this section to restore the proper information in the sysusages, sysdatabases, and sysdevices tables.

Restoring *sysdevices* with *disk reinit*

If you have added any database devices since the last dump—that is, if you have issued a disk init command—you must add each new device to sysdevices with disk reinit. If you saved scripts from your original disk init commands, use them to determine the parameters for disk reinit (including the original value of vstart). If the size you provide is too small, or if you use a different vstart value, you may corrupt your database.

If you did not save your disk init scripts, look at your most recent hard copy of sysdevices to determine some of the correct parameters for disk reinit. You will still need to know the value of vstart if you used a custom vstart in the original disk init command.

Table 29-1 describes the disk reinit parameters and their corresponding sysdevices data:

Table 29-1: Using sysdevices to determine disk reinit parameters

disk reinit parameter	*sysdevices* data	Notes
name	name	Use the same name, especially if you have any scripts that create or alter databases or add segments.
physname	*physname*	Must be full path to device.
vdevno	*low*/16777216	Not necessary to use the same value for *vdevno*, but be sure to use a value not already in use.
size	(*high* -*low*) +1	Extremely important to provide correct size information.

You can also obtain information on devices by reading the error log for *name*, *physname*, and *vdevno*, and using operating system commands to determine the size of the devices.

If you store your sybsystemprocs database on a separate physical device, be sure to include a disk reinit command for sybsystemprocs, if it is not listed in sysdevices.

After running disk reinit, compare your sysdevices table to the copy you made before running dataserver.

disk reinit can be run only from the master database and only by a System Administrator. Permission cannot be transferred to other users. Its syntax is:

```
disk reinit
    name = "device_name",
    physname = "physical_name",
    [vdevno = virtual_device_number,]
    size = number_of_blocks
     [, vstart = virtual_address,
    cntrltype = controller_number]
```

For more information on disk reinit, see the discussion of disk init in Chapter 16, “Initializing Database Devices,” or the *Reference Manual*.

Restoring *sysusages* and *sysdatabase* with *disk refit*

If you have added database devices or created or altered databases since the last database dump, use disk refit to rebuild the sysusages and sysdatabases tables.

disk refit can be run only from the master database and only by a System Administrator. Permission cannot be transferred to other users. Its syntax is:

```
disk refit
```

Adaptive Server shuts down after disk refit rebuilds the system tables. Examine the output while disk refit runs and during the shutdown process to determine whether any errors occurred.

Warning! Providing inaccurate information in the disk reinit command may lead to permanent corruption when you update your data. Be sure to check Adaptive Server with dbcc after running disk refit.

CHAPTER 30 Automatic Database Expansion

Automatic expansion of databases and devices lets you configure databases to expand automatically when they run out of space.

The automatic database expansion stored procedure sp_dbextend allows you to install thresholds that identify those devices with available space, and then appropriately alter the database—and the segment where the threshold was fired—on these devices.

Introduction

After you set up the database for automatic expansion, when a database grows to its free space threshold, internal mechanisms fire, increasing the size of the database by the amount of space your expansion policies specify. The automatic expansion process measures the amount of room left on all devices bound to the database. If there is sufficient room on the devices, the database continues to grow. If any devices are configured for expansion, those devices expand next. Finally, the database is expanded on those devices.

This automatic expansion process runs as a background task and generates informational messages in the server's error log about its progress.

Understanding disks, devices, databases, and segments

Figure 30-1 on page 972 shows the various layouts of physical resources that may exist in an Adaptive Server installation after a series of disk init, create database, and alter database operations. You can use this information to devise different plans of physical and logical space layout while testing stored procedures.

Raw disks can be partially occupied by a Sybase device. Device syb_device2 shows an entire raw disk fully occupied by a single Sybase device, on which multiple databases were created. On this raw disk (*/dev/vx/rdsk/sybase2/vol02*), there is still some empty space at the head of the device, which can happen when a database that initially occupied this space is subsequently dropped.

syb_device4 and syb_device5 show the layout of Sybase devices */agni4/syb_device4.dat* and */agni5/syb_device5.dat* on a file system disk, where the Sybase device occupies a portion of the disk, but there is still room for the device (an operating system file, for instance) to grow.

syb_device6 shows a Sybase file system disk that has fully occupied the entire available space on the physical disk, but still has unused space on the device. This space can be used to expand existing databases on this device.

These various devices illustrate database fragments for different databases. Each device for a particular database has one or more segments spanning that device.

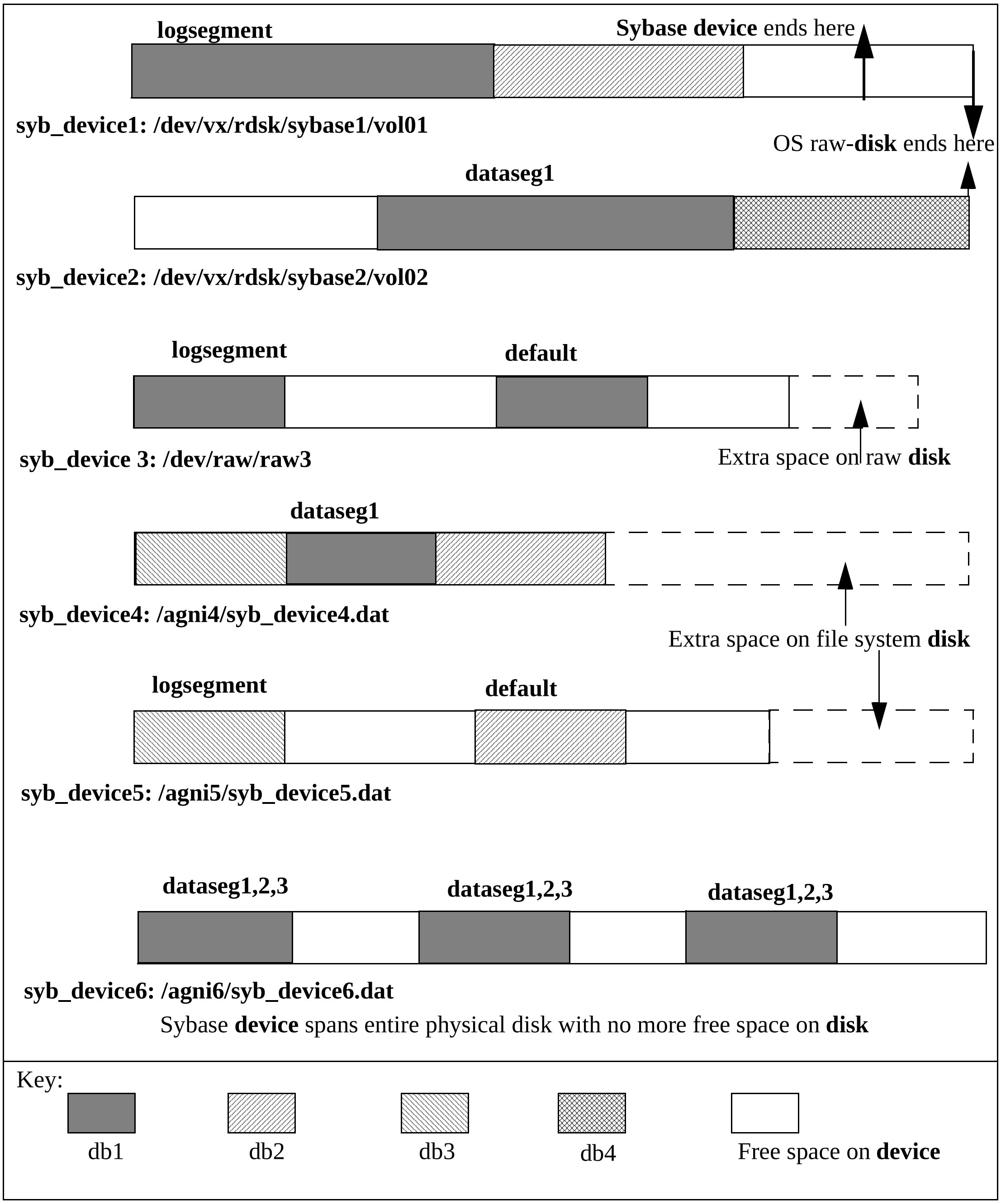
In syb_device6, */agni6/syb_device6.dat*, db1 spans three separate pieces of the device. That device also belongs to three different segments, data segments 1, 2, and 3. All three entries in sysusages for this database, db1, will appear with a segment map value that includes all three segments.

However, on the device syb_device3 on */dev/raw/raw3*, the database has two pieces of the device, and one of them is exclusively marked as for the log segment while the other is marked as for the default segment. Assuming that an initial create database command has already been executed, the following SQL commands can produce this result:

```
alter database db1 on syb_device3 = "30M"
alter database db1 log on syb_device3 = "10M" with
      override
```

The first alter database command creates the database piece of default segment, and the second one creates the database piece of logsegment, forcing an override even though both pieces are on the same device. Space is expanded on a database in individual segments.

Figure 30-1: Database and device layouts


logsegment
Sybase device ends here
syb_device1: /dev/vx/rdsk/sybase1/vol01
OS raw-disk ends here
dataseg1
syb_device2: /dev/vx/rdsk/sybase2/vol02
logsegment
default
syb_device 3: /dev/raw/raw3
Extra space on raw disk
dataseg1
syb_device4: /agni4/syb_device4.dat
Extra space on file system disk
logsegment
default
syb_device5: /agni5/syb_device5.dat
dataseg1,2,3
dataseg1,2,3
dataseg1,2,3
syb_device6: /agni6/syb_device6.dat
Sybase device spans entire physical disk with no more free space on disk
Key:
db1
db2
db3
db4
Free space on device

Threshold action procedures

Database expansion is performed by a set of threshold action procedures fired when free space crosses a threshold set for a segment. sp_dbextend is the interface for administering and managing the expansion process for a specified segment or device.

You can configure automatic expansion to run with server-wide default expansion policies, or you can customize it for individual segments in specified databases.You can install thresholds on key segments on which tables with critical data reside, allowing you a fine degree of control over how Adaptive Server meets the data space requirements for different kinds of tables. If your site has key tables with large volumes of inserts, these tables can be bound to specific segments, with site-specific rules for extending that segment. This enables you to avoid outages that can occur in a production environment with large loads on such key tables.

You cannot use the thresholds to shrink a database or its segment.

For more information on sp_dbextend, see the *Adaptive Server Reference Manual*.

Installing automatic database expansion procedures

Install the automatic expansion process using the *installdbextend* script, which loads rows into master.dbo.sysattributes describing defaults for automatic expansion in a database or in a device. The installation script also creates a control table in the model and tempdb databases.

If you are upgrading to Adaptive Server version 12.5.1 or later, you must install this script separately as part of your upgrade process.

v **Installing automatic database expansion**

1 Log in with sa_role permissions. In UNIX, *installdbextend* is located in the directory *$SYBASE/$SYBASE_ASE/scripts*. If you are running Windows NT, the location is *%SYBASE%/%SYABASE_ASE%/scripts*

2 Run *installdbextend* by entering:

```
isql -Usa -P -Sserver_name <$SYBASE/$SYBASE_ASE/scripts/installdbextend
```

The *installdbextend* installation script installs the family of threshold action procedures and sp_dbextend in the sybsystemprocs database.

Using the stored procedure

sp_dbextend allows you to customize the database and device expansion process, based on site-specific rules. Database administrators can configure the size or percentage by which each database and segment pair should be expanded, and the size or percentage by which each device should be expanded.

You can also limit the expansion of a database segment or device by specifying a maximum size beyond which no further expansion is possible.

You can use sp_dbexpand in a test mode, to simulate the expansion processes based on your chosen policies.

sp_dbextend provides ways to list the current settings and drop site-specific policy rules.

This information is stored as new attribute definitions in master.db.sysattributes.

Command options in the sp_dbextend interface

sp_dbextend has the following syntax:

```
sp_dbextend [ command [, arguments...] ]
```

where command is one of the options discussed below, and arguments specifies the database name, segment name, and amount of free space, among other things. For more information about sp_dbextend, see the *Reference Manual*.

The following command parameters define and specify user choices in sp_dbextend:

- set – sets the threshold at which a database and segment pair should fire. This command also allows you to specify the size and define the growth rate by which to expand the database and segment or device at each attempt, either in size values or as a percentage of the size of the device when the expansion is attempted.

You can also use set to specify the maximum allowable size for the database, set the site-specific policy rules for expanding a device, and set the maximum size for the device.

- clear – clears any previously set rules of expansion for a specified database and segment, or for a specified device.

  Using clear deletes the affected rows from master.db.systattributes, but automatic expansion is still in effect. Default rules for searching the devices a segment maps onto still apply when you consider where to expand the database, which device to expand, and in what order.

  For instance, if you issue clear for one device, only that device continues as a candidate for expansion under the default expansion rules. Other devices expand according to their individual device characteristics.

  To turn the automatic expansion feature completely off in a given database and the devices that database resides on, use the command 'clear', 'threshold'. This command executes sp_dropthreshold.

- modify – modifies site-specific policies, such as *growby* and *maxsize* in an existing entry for a database and segment. You can also use modify to modify system-supplied defaults.

  There is no support for modifying existing thresholds at which the database expansion threshold procedure fires. Use the existing interface sp_modifythreshold to make modifications.

- list – lists any existing rules for a specified database, segment, or device, and presents the data from master.db.sysattributes in a readable format.

  list allows you to view site-specific current policy rules in effect for each database, segment, or device previously configured by a set command. Databases, segments, and devices that do not appear in the output are subject to the default rules of expansion, which list also displays.

- listfull – lists fully the site-specific rules, including a comment column in the sysattributes table that displays a datetime stamp for when the rule was set and when it was last modified.

- check – examines current policies and verifies that they are consistent with the current space layout in each segment. If any policy settings appear redundant, ineffective, or incorrect, a warning message appears.

- simulate – simulates the database or device expansion schemes executed at runtime, according to the set of current policies implemented by the set command.

You can perform these cycles of execution as many times as you like, to study the way database and disk expansion operate and to determine what pieces expand, and by how much space. To perform the expansion, use the execute parameter.

- execute – performs the database and segment, or device, expansion using the current set of policies.This option performs the expansion process immediately, irrespective of the current free space in the specified segment.
- reload [defaults] – re initializes sysattributes with the system-supplied defaults for the *growby* and *maxsize* parameters, in all databases, segments, and devices, to revert to the original default behavior.

  For instance, you might make changes to system default rules using modify, and then run a simulation with simulate. Running reload [defaults] deletes any existing rows describing system default behavior and loads new rows. It does not change existing rows defining user-specified policies.
- help – provides help in using the procedure, or specific help information for each command.
- trace – turns tracing facility on or off, in order to trace through the procedure execution.
- enable / disable – enables or disables the automatic expansion procedures on a specified database segment or device.
- who – shows any active expansion processes running currently. ’<spid>’ restricts the output for a particular spid.

If you provide no argument, sp_dbextend defaults to help. For more information on sp_dbextend, see the *Reference Manual*.

Note The automatic expansion procedure does not create new devices; it only alters the size of the database and segment on existing devices to which the segment currently maps.

Dropping the threshold action procedure

To discontinue threshold action procedure, clear the threshold using the sp_dropthreshold, or use sp_dbextend with the *clear* option. For information about sp_dropthreshold, see the *Reference Manual*.

Testing the process with *sp_dbextend*

You can use the check parameter to validate the current settings of various thresholds. For instance, check warns you if multiple segments share the same set of devices and both segments are set for automatic expansion, or if the threshold currently set to trigger automatic expansion on the logsegment is too close to the current last-chance threshold for the logsegment. In this situation, the automatic threshold does not fire, and check reports a warning.

sp_dbextend offers a powerful simulation mode that any user with sa_role permissions can use to simulate the execution of the top-level threshold action procedure.

- To define expansion policies for the logsegment in the pubs2 database:

```
sp_dbextend ’set’, ’database’, pubs2, logsegment,’3M’

sp_dbextend ’set’, ’threshold’, pubs2, logsegment, ’1M’
```

- To simulate expansion for these policies:

  ```
  sp_dbextend ’simulate’, pubs2, logsegment
  ------------------------------
  ```

  Messages from the server follow this input.

  The following outputs show the series of database and disk expansions that would occur if the threshold on database pubs2 segment logsegment fired once:

```
sp_dbextend 'simulate', pubs2, logsegment
---------------
NO REAL WORK WILL BE DONE.
Simulate database / device expansion in a dry-run mode 1 time(s).
These are the series of database/device expansions that would have
happened if the threshold on database'pubs2',
segment 'logsegment' were to fire 1 time(s).

Threshold fires: Iteration: 1.
=============================

Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'.

Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M'
-- Segment: logsegment

Database 'pubs2' was altered by total size '3M' for
```

```
segment 'logsegment'.
Summary of device/database sizes after 1 simulated extensions:
==================================================
devicename initial size final size
---------- ------------ ----------
pubs2_data 20.0M        20.0M

(1 row affected)

Database 'pubs2', segment 'logsegment' would be altered from an initial
size of '4M' by '3M' for a resultant total size of '7M'.

To actually expand the database manually for this threshold, issue:
sp_dbextend 'execute', 'pubs2','logsegment', '1'

(return status = 0)
```

To expand the database manually for this threshold, execute:

```
sp_dbextend ’execute’, ’pubs2’, ’logsegment’
--------------
```

This output shows that if the threshold fires at this level an alter database command operates on the pubs2_data device for the logsegment:

```
sp_dbextend 'execute', pubs2, logsegment

Threshold fires: Iteration: 1.
================================
Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'. Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M' -- Segment: logsegment
Extending database by 1536 pages (3.0 megabytes) on disk pubs2_data
Warning: The database 'pubs2' is using an unsafe virtual device
'pubs2_data'. The recovery of this database can not be guaranteed.

Warning: Using ALTER DATABASE to extend the log segment will cause user
thresholds on the log segment within 128 pages of the last chance
threshold to be disabled.

Database 'pubs2' was altered by total size '3M' for segment 'logsegment'.

(return status = 0)
```

- To simulate what would actually happen if the threshold fired <*n*> times in succession on a particular segment, issue the same command, specifying the number of iterations:

```
sp_dbextend 'simulate', pubs2, logsegment, 5
-----------------
```

The following example shows how to expand this database five times:

```
sp_dbextend 'execute', 'pubs2', 'logsegment', 5
------------------
```

The output this provides shows that firing the threshold five times in succession puts the database through a series of alter database operations, followed by one or more disk resize operations and, finally, an alter database on the specified device.

Setting up the pubs2 database for automatic expansion

To set up different segments in the pubs2 database for automatic expansion, follow this procedure. Not all these steps are mandatory. For example, you may choose not to set *growby* or *maxsize* for individual devices, and to use the system default policies only for the devices.

v **Setting up pubs2**

1 Create the database. Enter:

```
create database pubs2 on pubs2_data = "10m" log on pubs2_log = "5m"
```

2 Set the *growby* and *maxsize* policies for the pubs2_data device at 10MB and 512MB respectively. You can be in any database to set these policies; you need not be in the specified database. Enter:

```
exec sp_dbextend 'set', 'device', pubs2_data, '10m', '512m'
```

3 The system default *growby* policy is 10% for devices. Rather than set new policies for the pubs2_log device, you can modify this system default, choosing an appropriate *growby* value. The pubs2_log then expands at this rate. Enter:

```
exec sp_dbextend 'modify', 'device', 'default', 'growby', '3m'
```

4 Set the growby rate for the default segment, but do not specify a maximum size. Enter:

```
exec sp_dbextend 'set', 'database', pubs2, 'default', '5m'
```

The *growby* rate on the default segment may be different from that on the devices where the segment resides. *growby* controls the segment's expansion rate when it is running out of free space, and is used only when you expand the segment.

5 Set the *growby* and *maxsize* variables for the logsegment:

```
 exec sp_dbextend 'set', 'database', pubs2, 'logsegment', '4m', '100m'
```

6 Examine the policies established for various segments in the pubs2 database:

```
exec sp_dbextend 'list', 'database', pubs2
```

7 Examine the policies in the various devices that pubs2 spans. The pattern specifier for *devicename* ("%") picks up all these devices:

```
exec sp_dbextend 'list', 'device', "pubs2%"
```

8 Install the expansion threshold for the default and logsegments segments in pubs2. This sets up and enables the expansion process, and allows you to choose the free space threshold at which to trigger the expansion process. Enter:

```
 use pubs2
----------------------------------------------------------------

    exec sp_dbextend 'set', 'threshold', pubs2, 'default',   '4m'
    exec sp_dbextend   'set', 'threshold',  pubs2,  'logsegment', '3m'
```

9 Examine the thresholds installed by the commands above.

```
exec sp_dbextend list, 'threshold'

segment name free pages  free pages (KB) threshold procedure status
------------ ----------- --------------- ------------------- -----------
default           2048    4096             sp_dbxt_extend_db enabled
logsegment         160  320          sp_thresholdaction lastchance
logsegment        1536  3072         sp_dbxt_extend_db      enabled
Log segment free space currently is 2548 logical pages (5096K).
(1 row affected, return status = 0)
```

In this output, sp_dbxt_extend_db is the threshold procedure that drives the expansion process at runtime. The expansion thresholds are currently enabled on both the default and logsegment segments.

10 Use simulate to see the expansion:

```
exec sp_dbextend 'simulate', pubs2, logsegment
exec sp_dbextend 'simulate', pubs2, 'default', '5'
```

11 Use modify to change the policy if necessary:

```
exec sp_dbextend 'modify', 'database', pubs2, logsegment, 'growby','10m'
```

12 To disable expansion temporarily on a particular segment, use disable:

```
exec sp_dbextend 'disable', 'database', pubs2, logsegment
```

13 Examine the state of the expansion policy on databases and devices:

```
exec sp_dbextend list, 'database'
name        segment    item    value status
----------- ---------- ------- ----- --------
server-wide (n/a)      (n/a)   (n/a) enabled
default     (all)      growby  10%   enabled
pubs2       default    growby  5m    enabled
pubs2       logsegment growby  10m   disabled
pubs2       logsegment maxsize 100m  disabled
(1 row affected, return status = 0)
```

The status disabled indicates that the expansion process is currently disabled on the logsegment in pubs2.

```
exec sp_dbextend list, 'device'
name           segment item   value status
-------------- ------- ------ ----- -------
server-wide    (n/a)   (n/a)  (n/a) enabled
default        (n/a)   growby 3m    enabled
mypubs2_data_0 (n/a)   growby 10m   enabled
mypubs2_data_1 (n/a)   growby 100m  enabled
mypubs2_log_1  (n/a)   growby 20m   enabled
mypubs2_log_2  (n/a)   growby 30m   enabled
(1 row affected, return status = 0)
```

14 Use enable to reenable the expansion process:

```
exec sp_dbextend 'enable', 'database', pubs2, logsegment
```

Changes to existing procedures and system tables

No new system tables are created in this feature. Policy rules for automatic database expansion are stored in the master.dbo.sysattributes database.

Restrictions and limitations

- When threshold procedures are installed on multiple segments in one or more databases, the expansion is performed in the order in which the thresholds fire. If abort tran on log full is off for the logsegment, tasks wait until the threshold procedure for the logsegment is scheduled to alter the database.
- In non-log segments, tasks continue to process even after the free space threshold is crossed, while the threshold procedure remains in queue. This can cause "out of space" errors in data segments. Design your thresholds to have sufficient space in the database for the task to complete.
- If many threshold procedures fire at the same time, the procedure cache may become overloaded. This is more likely to occur in installations with large numbers of databases, many segments, and many threshold action procedures installed.
- If the space in the tempdb is very low, and other operations need tempdb resources, the threshold procedures may fail even while trying to correct the situation. Make sure that threshold procedures in tempdb are installed with sufficiently large amounts of free space, at least 2MB, to avoid this problem.

You may need to change your dump and load procedures to manage site-specific policies that determine how databases and devices are expanded.

Dumping a database does not transport information stored in master.db.sysattributes, so if you use dump and load as a way to migrate databases from a source server to a target server, you must manually migrate any site-specific policies encoded as data in the sysattributes database. There are two possible workarounds:

- Using bcp out from a view defined on master.dbo.sysattributes for entries with class number 19, you can manually extract the data from master.dbo.sysattributes, then use bcp in to load the data into the target server. This requires that both databases across the two servers have the same segment IDs.
- You can also use the ddlgen feature of Sybase Central to regenerate the sp_dbextend set invocations necessary to re-create your policy rules, by running the ddlgen script at the target server. However, renamed logical devices across servers cannot be managed by the ddlgen procedure. You must rename the devices manually at the target server.

The following restrictions do not cause failure.

- You can install a threshold action on a non-log segment when the database has the option 'no free space acctg' turned on. This option means only that no database expansion is performed, since threshold actions are not fired with this option is off. Leaving this option on generates a warning message.
- Sybase recommends that you periodically dump the master database if expansion occurs, so that you can re-create the master database in case of failure after several expansions.
- Sybase recommends that you do not install these generic threshold procedures on any system databases, particularly the master database, as modifying space usage in the master database requires special treatment.
- You cannot use thresholds to shrink a database or segment.

CHAPTER 31 Managing Free Space with Thresholds

When you create or alter a database, you allocate a finite amount of space for its data and log segments. As you create objects and insert data, the amount of free space in the database decreases.

This chapter explains how to use thresholds to monitor the amount of free space in a database segment.

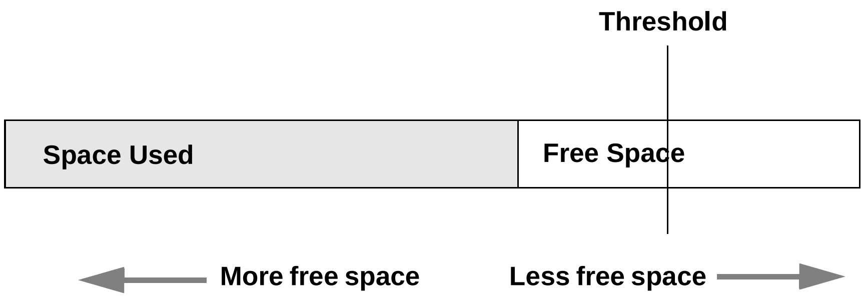
Monitoring free space with the last-chance threshold

All databases have a **last-chance threshold**, including master. The threshold is an estimate of the number of free log pages that are required to back up the transaction log. As you allocate more space to the log segment, Adaptive Server automatically adjusts the last-chance threshold.

When the amount of free space in the log segment falls below the last-chance threshold, Adaptive Server automatically executes a special stored procedure called sp_thresholdaction. (You can specify a different last-chance threshold procedure with sp_modifythreshold.)

Figure 31-1 illustrates a log segment with a last-chance threshold. The shaded area represents log space that has already been used; the unshaded area represents free log space. The last-chance threshold has not yet been crossed.

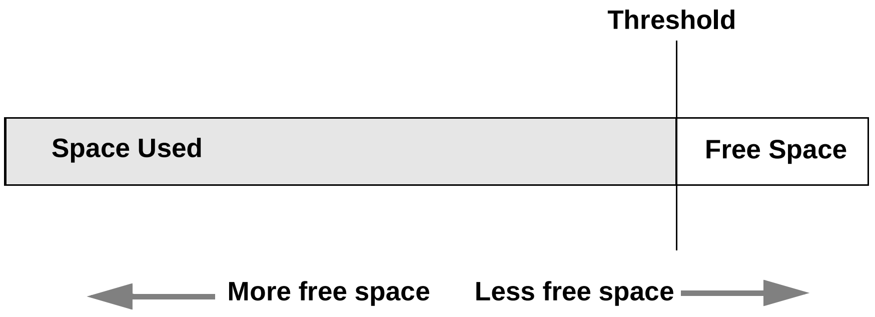
Figure 31-1: Log segment with a last-chance threshold



Crossing the threshold

As users execute transactions, the amount of free log space decreases. When the amount of free space crosses the last-chance threshold, Adaptive Server executes sp_thresholdaction:

Figure 31-2: Executing sp_thresholdaction when the last-chance threshold is reached



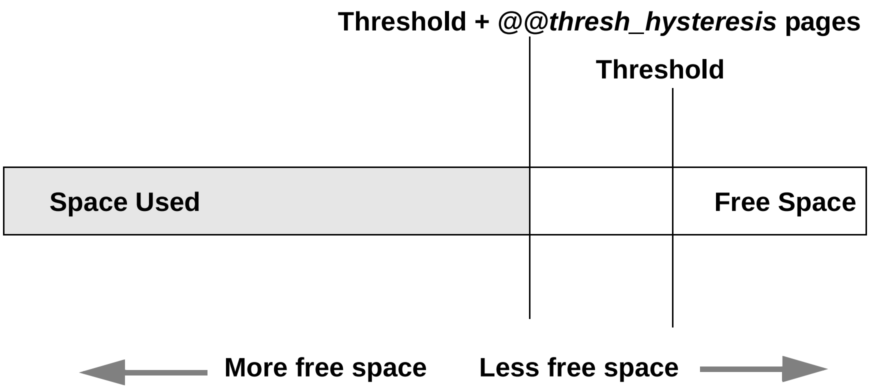
Controlling how often *sp_thresholdaction* executes

Adaptive Server uses a *hysteresis value*, the global variable *@@thresh_hysteresis*, to control how sensitive thresholds are to variations in free space.

A threshold is deactivated after it executes its procedure, and remains inactive until the amount of free space in the segment rises *@@thresh_hysteresis* pages above the threshold. This prevents thresholds from executing their procedures repeatedly in response to minor fluctuations in free space. You cannot change the value of *@@thresh_hysteresis*.

For example, when the threshold in Figure 31-2 executes sp_thresholdaction, it is deactivated. In Figure 31-3, the threshold is reactivated when the amount of free space increases by the value of *@@thresh_hysteresis*:

Figure 31-3: Free space must rise by @@thresh_hysteresis to reactivate threshold



Rollback records and the last-chance threshold

Adaptive Server version 11.9 and later include **rollback records** in the transaction logs. Rollback records are logged whenever a transaction is rolled back. Servers save enough space to log a rollback record for every update belonging to an open transaction. If a transaction completes successfully, no rollback records are logged, and the space reserved for them is released.

In long-running transactions, rollback records can reserve large amounts of space.

To check the space used by the syslogs, run the sp_spaceused:

```
sp_spaceused syslogs
```

The output is:

```
name           total_pages    free_pages     used_pages     reserved_pages
-------------- -------------- -------------- -------------- --------------
syslogs                  5632           1179           3783            670
```

The dbcc checktable(syslogs) produce a similar output as:

```
Checking syslogs: Logical pagesize is 2048 bytes
The total number of data pages in this table is 3761.
*** NOTICE:  Space used on the log segment is 3783 pages (7.39 Mbytes), 67.17%.
> *** NOTICE:  Space reserved on the log segment is 670 pages (1.31 Mbytes),
11.90%.
*** NOTICE:  Space free on the log segment is 1179 pages (2.30 Mbytes), 20.93%.
```

If the last chance threshold for the transaction log fires when it seems to have sufficient space, it may be the space reserved for rollbacks that is causing the problem. See "Determining the current space for rollback records" on page 989 for more information.

Calculating the space for rollback records

To calculate the increased amount of space to add to a transaction log to accommodate rollback records, estimate:

- The number of update records in the transaction log that are likely to belong to already rolled-back transactions
- The maximum number of update records in the transaction log that are likely to belong to open transactions at any one time

Update records are the records that change the timestamp value. They include changes to datapages, indexpages, allocation pages, and so on.

Each rollback record requires approximately 60 bytes of space, or 3 one hundredths of a page. Thus, the calculation for including rollback records (RRs) in the transaction log is:

Added space, in pages = (logged RRs + # open updates) X 3/100.

You may also want to add log space to compensate for the effects of rollback records on the last-chance threshold and on user-defined thresholds, as described in the following sections.

Using lct_admin to determine the free log space

Use the logsegment_freepages to determine the amount of free space your dedicated log segment has. The syntax is:

lct_admin("logsegment_freepages", database_id)

To see the amount of free pages for the pubs2 database log segment:

```
select lct_admnin("logsegment_freepages", 4)
---------------------
                      79
```

Determining the current space for rollback records

To determine the number of pages a database currently reserved for rollbacks, issue lct_admin with the reserved_for_rollbacks parameter. The partial syntax for lct_admin is:

select lct_admin ("reserved_for_rollbacks", *dbid*)

The number of pages returned are the number reserved, but not yet allocated, for rollback records.

For example, to determine the number of pages reserved for rollbacks in the pubs2 database (which has a pubid of 5), issue the following:

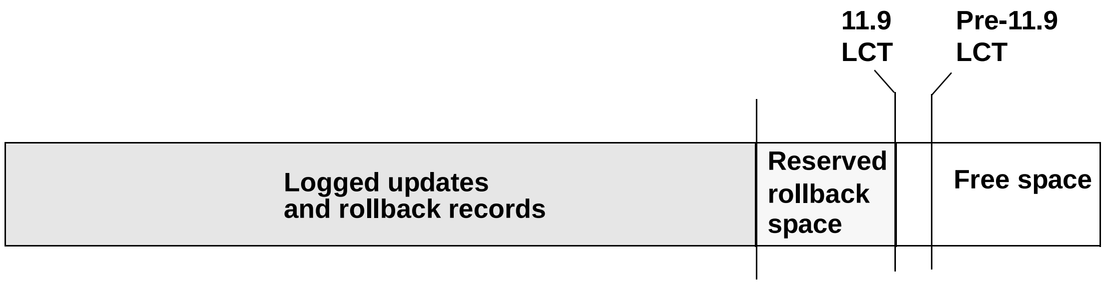
```
select lct_admin("reserved_for_rollbacks", 5)
```

See the *Reference Manual* for more information about lct_admin.

Effect of rollback records on the last-chance threshold

Adaptive Servers that use rollback records must reserve additional room for the last-chance threshold. The last-chance threshold (“LCT”) is also likely to be reached sooner because of the space used by already logged rollback records and the space reserved against open transactions for potential rollback records. Figure 31-4 illustrates how space is used in a transaction log with rollback records:

Figure 31-4: Space used in log with rollback records



In Figure 31-4, in the 11.9 Adaptive Server, log space is occupied by logged rollback records for closed transactions that did not complete successfully. In addition, space is reserved for rollback records that may need to be logged, if any of the currently open transactions do not complete successfully. Together, the space for logged rollback records and for potential rollback records is likely to be considerably greater than the extra space for the last-chance threshold. Consequently, transaction logs that use rollback records reach the last-chance threshold significantly sooner than transaction logs that do not use rollback records, even if the size of the transaction log is not increased.

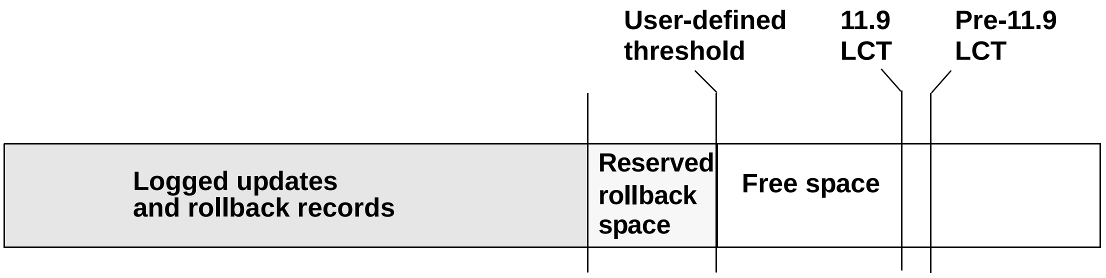
In general, about 18 percent more log space is reserved for the last-chance threshold in 11.9 and later Adaptive Servers than in earlier versions. For example, for a transaction log of 5000 pages, version 11.5 reserves 264 pages and version 11.9 reserves 312 pages. This is an increase of 18.18 percent between version 11.5 and 11.9.

User-defined thresholds

Because rollback records occupy extra space in the transaction log, there is less free space after the user-defined threshold for completing a dump than in versions of Adaptive Server that do not use rollback records (See Figure 31-4). However, the loss of space for a dump because of the increased last-chance threshold is likely to be more than compensated for by the space reserved for rollback records for open transactions.

Upgrading to version that uses rollback records affects user-defined thresholds similarly to the way it effects last-chance thresholds. Figure 31-5 illustrates the effect of upgrading to an Adaptive Server using rollback records on a user-defined threshold:

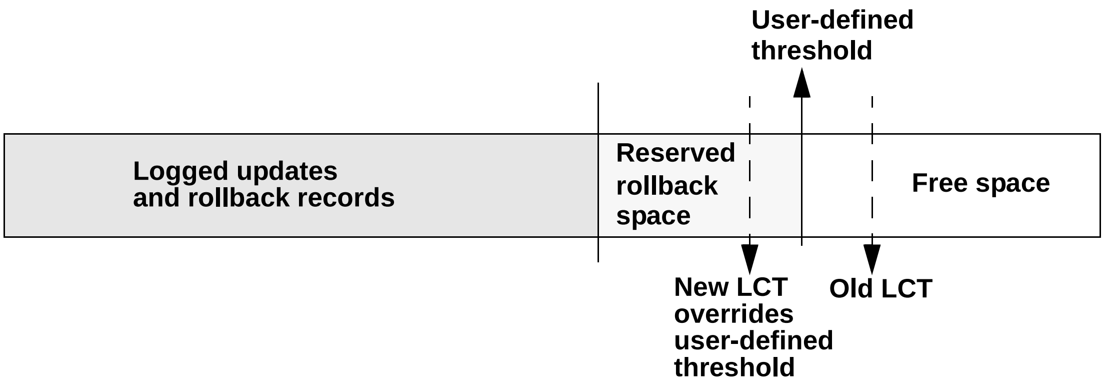
Figure 31-5: Effect of upgrading on user-defined thresholds



A user-defined threshold such as the one in Figure 31-5 is often used to initiate a dump transaction. The threshold is set so there is enough room to complete the dump before the last-chance threshold is reached and all open transactions in the log are suspended.

In databases that use mixed log and data, the last-chance threshold moves dynamically, and its value can be automatically configured to be less than the user-defined threshold. If this happens, the user-defined threshold is disabled, and the last chance threshold fires before the user-defined threshold is reached, as shown in Figure 31-6:

Figure 31-6: LCT firing before user-defined threshold


User-defined threshold
Logged updates and rollback records
Reserved rollback space
Free space
New LCT overrides user-defined threshold
Old LCT

The user-defined threshold is re-enabled if the value of last-chance threshold is configured to be greater than the user-defined threshold (for example, if the last chance threshold is reconfigured for the value of “Old LCT” in Figure 31-6).

In databases with a separate log segment, the log has a dedicated amount of space and the last-chance threshold is static. The user-defined threshold is not affected by the last-chance threshold.

Last-chance threshold and user log caches for shared log and data segments

Every database in an Adaptive Server has a last-chance threshold and all databases allow transactions to be buffered in a user log cache. When you initially create a database with shared log and data segments, its last-chance threshold is based on the size of the model database. As soon as data is added and logging activity begins, the last-chance threshold is recalculated dynamically, based on available space and currently open transactions. The last-chance threshold of a database with separate log and data segments is based on the size of the log segment and does not vary dynamically.

To get the current last-chance threshold of any database, you can use lct_admin with the reserve parameter and a specification of 0 log pages:

```
select lct_admin("reserve",0)
```

The last-chance threshold for a database is stored in the systhresholds table and is also accessible through sp_helpthreshold. However, note that:

- sp_helpthreshold returns user-defined thresholds and other data, as well as an up-to-date value for the last-chance threshold. Using lct_admin is simpler if you need only the current last-chance threshold. Either of these values produce the most current value for the last-chance threshold.
- For a database with shared log and data segments, the last-chance threshold value in systhresholds may *not* be the current last-chance threshold value.

Reaching last-chance threshold suspends transactions

The default behavior of Adaptive Server is to suspend open transactions until additional log space is created. Transactions suspended because of the last-chance threshold can be terminated using the abort parameter of the lct_admin system function, described in "Using lct_admin abort to abort suspended transactions," below. For information about configuring Adaptive Server to automatically abort suspended processes, see "Automatically aborting or suspending processes" on page 995.

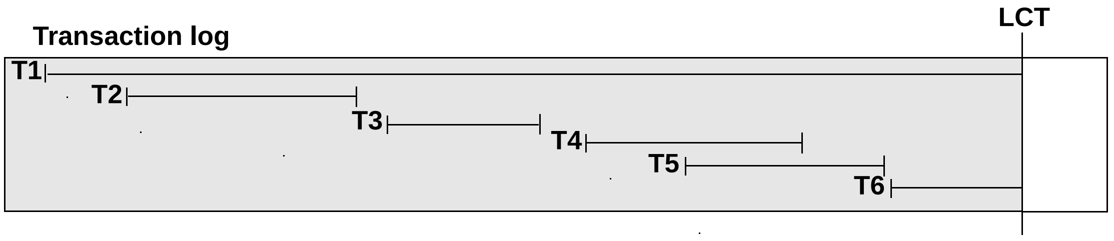
Using *lct_admin abort* to abort suspended transactions

When the transaction log reaches the last-chance threshold, all becomes made available. Typically, space is created by dumping the transaction log, since this removes committed transactions from the beginning of the log. However, if one or more transactions at the beginning of the log is still open, it prevents a dump of the transaction log.

Use lct_admin abort to terminate suspended transactions that are preventing a transaction log dump. Since terminating a transaction

closes it, this allows the dump to proceed. Figure 31-7 illustrates a possible scenario for using lct_admin abort:

Figure 31-7: Example of when to use of lct_admin abort



In Figure 31-7, a transaction log has reached its LCT, and open transactions T1 and T6 are suspended. Because T1 is at the beginning of the log, it prevents a dump from removing closed transactions T3 through T5 and creating space for continued logging. Terminating T1 with lct_admin abort allows you to close T1 so that a dump can clear transactions T1 through T5 from the log.

lct_admin abort replaces lct_admin unsuspend.

lct_admin abort syntax

The syntax for lct_admin abort is:

lct_admin("abort", {*process_id* [, *database_id]}*)

Before you can abort a transaction, you must first determine its ID. See "Getting the process ID for the oldest open transaction," below for information about determining the transaction's pid.

To terminate the oldest transaction, enter the process ID (spid) of the process that initiated the transaction. This also terminates any other suspended transactions in the log that belong to the specified process.

For example, if process 83 holds the oldest open transaction in a suspended log, and you want to terminate the transaction, enter:

```
select lct_admin("abort", 83)
```

This also terminates any other open transactions belonging to process 83 in the same transaction log.

To terminate all open transactions in the log, enter:

```
select lct_admin("abort", 0, 12)
```

Getting the process ID for the oldest open transaction

Use the following query to find the spid of the oldest open transaction in a transaction log that has reached its last-chance threshold:

```
use master
go
select dbid, spid from syslogshold
where dbid = db_id("name_of_database")
```

For example, to find the oldest running transaction on the pubs2 database:

```
select dbid, spid from syslogshold
where dbid = db_id ("pubs2")
dbid   spid
------ ------
     7      1
```

Using *alter database* when the master database reaches the last-chance threshold

When the last-chance threshold on the master database is reached, you can use alter database to add space to the master database's transaction log. This allows more activity in the server by causing suspended transactions in the log to become active. However, while the master transaction log is at its last-chance threshold, you cannot use alter database to make changes in other databases. Thus, if both master and another database reach their last-chance thresholds, you would first need to use alter database to add log space to the master database, and then use it again to add log space to the second database.

Automatically aborting or suspending processes

By design, the last-chance threshold allows enough free log space to record a dump transaction command. There may not be enough room to record additional user transactions against the database.

When the last-chance threshold is crossed, Adaptive Server suspends user processes and displays the message:

```
Space available in the log segment has fallen critically
low in database 'mydb'. All future modifications to this
database will be suspended until the log is successfully
dumped and space becomes available.
```

Only commands that are not recorded in the transaction log (select or readtext) and commands that might be necessary to free additional log space (dump transaction, dump database, and alter database) can be executed.

Using *abort tran on log full* to abort transactions

To configure the last-chance threshold to automatically abort open transactions, rather than suspend them:

```
sp_dboption database_name "abort tran on log full", true
```

If you upgrade from a previous version of Adaptive Server, the newly upgraded server retains the abort tran on log full setting.

Waking suspended processes

After dump transaction frees sufficient log space, suspended processes automatically awaken and complete. If writetext or select into has resulted in unlogged changes to the database since the last backup, the last-chance threshold procedure cannot execute dump transaction. When this occurs, make a copy of the database with dump database, then truncate the log with dump transaction.

If this does not free enough space to awaken the suspended processes, you may need to increase the size of the transaction log. Use the log on option of alter database to allocate additional log space.

As a last resort, System Administrators can use the sp_who command to determine which processes are in a log suspend status and then use the kill command to kill the sleeping process.

Adding, changing, and deleting thresholds

The Database Owner or System Administrator can create additional thresholds to monitor free space on any segment in the database. Additional thresholds are called **free-space thresholds**. Each database can have up to 256 thresholds, including the last-chance threshold.

sp_addthreshold, sp_modifythreshold, and sp_dropthreshold allow you to create, change, and delete thresholds. To prevent users from accidentally affecting thresholds in the wrong database, these procedures require that you specify the name of the current database.

Displaying information about existing thresholds

Use sp_helpthreshold to get information about all thresholds in a database. Use sp_helpthreshold *segment_name* to get information about the thresholds on a particular segment.

The following example displays information about the thresholds on the database's default segment. Since "default" is a reserved word, you must enclose it in quotation marks. The output of sp_helpthreshold shows that there is one threshold on this segment set at 200 pages. The 0 in the "last chance" column indicates that this is a free-space threshold instead of a last-chance threshold:

```
    sp_helpthreshold "default"
```

```
segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   -------------------
default        200                     0   space_dataseg

(1 row affected, return status = 0)
```

Thresholds and system tables

The system table systhresholds holds information about thresholds. sp_helpthreshold uses this table to provide its information. In addition to information about segment name, free page, last-chance status, and the name of the threshold procedure, the table also records the server user ID of the user who created the threshold and the roles had at the moment the threshold was created.

Adaptive Server gets information about how much free space is remaining in a segment—and whether to activate a threshold—from the built-in system function curunreservedpgs().

Adding a free-space threshold

Use sp_addthreshold to create free-space thresholds. Its syntax is:

sp_addthreshold *dbname*, *segname*, *free_space*, *proc_name*

The *dbname* must specify the name of the current database. The remaining parameters specify the segment whose free space is being monitored, the size of the threshold in database pages, and the name of a stored procedure.

When the amount of free space on the segment falls below the threshold, an internal Adaptive Server process executes the associated procedure. This process has the permissions of the user who created the threshold when he or she executed sp_addthreshold, less any permissions that have since been revoked.

Thresholds can execute a procedure in the same database, in another user database, in sybsystemprocs, or in master. They can also call a remote procedure on an Open Server. sp_addthreshold does not verify that the threshold procedure exists when you create the threshold.

Changing a free-space threshold

Use sp_modifythreshold to associate a free-space threshold with a new threshold procedure, free-space value, or segment. sp_modifythreshold drops the existing threshold and creates a new one in its place. Its syntax is:

sp_modifythreshold *dbname* , *segname* , *free_space*
[, *new_proc_name* [, *new_free_space*
[, *new_segname*]]]

where *dbname* is the name of the current database, and *segname* and *free_space* identify the threshold that you want to change.

For example, to execute a threshold procedure when free space on the segment falls below 175 pages rather than below 200 pages, enter:

```
sp_modifythreshold mydb, "default", 200, NULL, 175
```

In this example, NULL acts as a placeholder so that *new_free_space* falls in the correct place in the parameter list. The name of the threshold procedure is not changed.

The person who modifies the threshold becomes the new threshold owner. When the amount of free space on the segment falls below the threshold, Adaptive Server executes the threshold procedure with the owner's permissions at the time he or she executed sp_modifythreshold, less any permissions that have since been revoked.

Specifying a new last-chance threshold procedure

You can use sp_modifythreshold to change the name of the procedure associated with the last-chance threshold. You *cannot* use it to change the amount of free space or the segment name for the last-chance threshold.

sp_modifythreshold requires that you specify the number of free pages associated with the last-chance threshold. Use sp_helpthreshold to determine this value.

The following example displays information about the last-chance threshold, and then specifies a new procedure, sp_new_thresh_proc, to execute when the threshold is crossed:

```
sp_helpthreshold logsegment
```

```
segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   -------------------
logsegment     40                      1   sp_thresholdaction

(1 row affected, return status = 0)
```

```
sp_modifythreshold mydb, logsegment, 40,
sp_new_thresh_proc
```

Dropping a threshold

Use sp_dropthreshold to remove a free-space threshold from a segment. Its syntax is:

sp_dropthreshold *dbame*, *segname*, *free_space*

The *dbname* must specify the name of the current database. You must specify both the segment name and the number of free pages, since there can be several thresholds on a particular segment. For example:

```
sp_dropthreshold mydb, "default", 200
```

Creating a free-space threshold for the log segment

When the last-chance threshold is crossed, all transactions are aborted or suspended until sufficient log space is freed. In a production environment, this can have a heavy impact on users. Adding a correctly placed free-space threshold on your log segment can minimize the chances of crossing the last-chance threshold.

The additional threshold should dump the transaction log often enough that the last-chance threshold is rarely crossed. It should not dump it so often that restoring the database requires the loading of too many tapes.

This section helps you determine the best place for a second log threshold. It starts by adding a threshold with a *free_space* value set at 45 percent of log size and adjusts this threshold based on space usage at your site.

Adding a log threshold at 45 percent of log size

Use the following procedure to add a log threshold with a *free_space* value set at 45 percent of log size.

1. Determine the log size in pages:

   ```
   select sum(size)
   from master..sysusages
   where dbid = db_id("database_name")
   and (segmap & 4) = 4
   ```

2. Use sp_addthreshold to add a new threshold with a *free_space* value set at 45 percent. For example, if the log's capacity is 2048 pages, add a threshold with a *free_space* value of 922 pages:

   ```
   sp_addthreshold mydb, logsegment, 922, thresh_proc
   ```

3. Create a simple threshold procedure that dumps the transaction log to the appropriate devices. For more information about creating threshold procedures, see "Creating threshold procedures" on page 1004.

Testing and adjusting the new threshold

Use dump transaction to make sure your transaction log is less than 55 percent full. Then use the following procedure to test the new threshold:

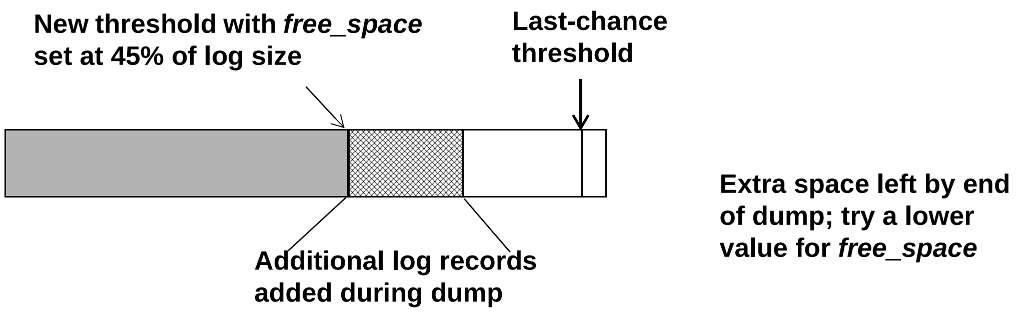
1. Fill the transaction log by simulating routine user action. Use automated scripts that perform typical transactions at the projected rate.

   When the 45 percent free-space threshold is crossed, your threshold procedure will dump the transaction log. Since this is not a last-chance threshold, transactions will not be suspended or aborted; the log will continue to grow during the dump.

2. While the dump is in progress, use sp_helpsegment to monitor space usage on the log segment. Record the maximum size of the transaction log just before the dump completes.

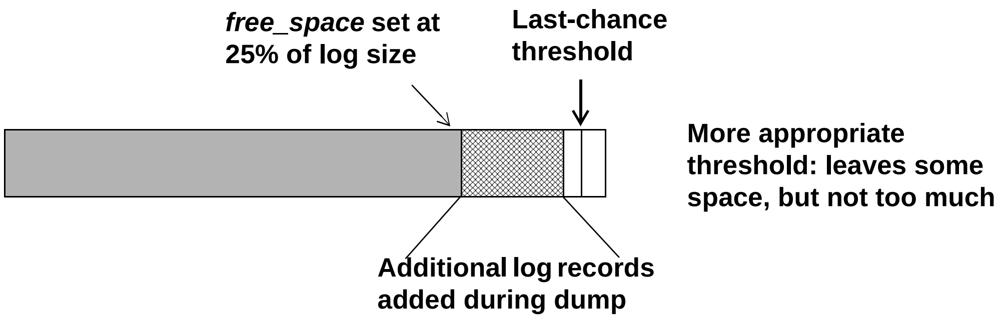
3 If considerable space was left in the log when the dump completed, you may not need to dump the transaction log so soon, as shown in Figure 31-8:

Figure 31-8: Transaction log with additional threshold at 45 percent


New threshold with *free_space* set at 45% of log size
Last-chance threshold
Extra space left by end of dump; try a lower value for *free_space*
Additional log records added during dump

Try waiting until only 25 percent of log space remains, as shown in Figure 31-9:

Figure 31-9: Moving threshold leaves less free space after dump

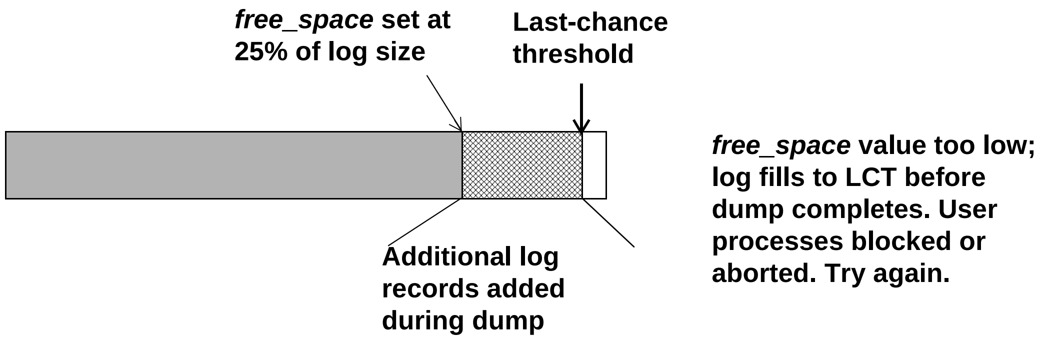

free_space set at 25% of log size
Last-chance threshold
More appropriate threshold: leaves some space, but not too much
Additional log records added during dump

Use sp_modifythreshold to adjust the *free_space* value to 25 percent of the log size. For example:

```
sp_modifythreshold mydb, logsegment, 512,
    thresh_proc
```

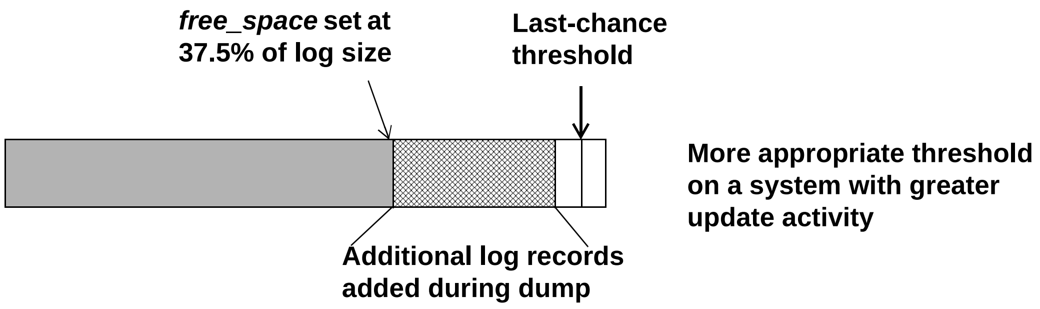
4 Dump the transaction log and test the new *free_space* value. If the last-chance threshold is crossed before the dump completes, you are not beginning the dump transaction soon enough, as shown in Figure 31-10:

Figure 31-10: Additional log threshold does not begin dump early enough


free_space set at 25% of log size
Last-chance threshold
free_space value too low; log fills to LCT before dump completes. User processes blocked or aborted. Try again.
Additional log records added during dump

25 percent free space is not enough. Try initiating the dump transaction when the log has 37.5 percent free space, as shown in Figure 31-11:

Figure 31-11: Moving threshold leaves enough free space to complete dump


free_space set at 37.5% of log size
Last-chance threshold
More appropriate threshold on a system with greater update activity
Additional log records added during dump

Use sp_modifythreshold to change the *free_space* value to 37.5 percent of log capacity. For example:

```
sp_modifythreshold mydb, logsegment, 768,
    thresh_proc
```

Creating additional thresholds on other segments

You can create free-space thresholds on data segments as well as on log segments. For example, you might create a free-space threshold on the default segment used to store tables and indexes. You would also create an associated stored procedure to print messages in your error log when space on the default segment falls below this threshold. If you monitor the error log for these messages, you can add space to the database device before your users encounter problems.

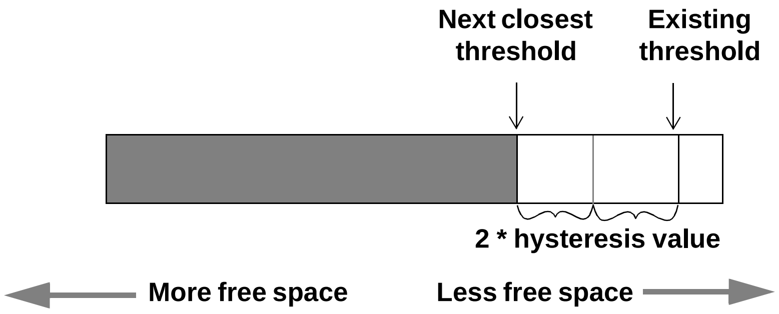
The following example creates a free-space threshold on the default segment of mydb. When the free space on this segment falls below 200 pages, Adaptive Server executes the procedure space_dataseg:

```
sp_addthreshold mydb, "default", 200, space_dataseg
```

Determining threshold placement

Each new threshold must be at least twice the *@@thresh_hysteresis* value from the next closest threshold, as shown in Figure 31-12:

Figure 31-12: Determining where to place a threshold



To see the hysteresis value for a database, use:

```
select @@thresh_hysteresis
```

In this example, a segment has a threshold set at 100 pages, and the hysteresis value for the database is 64 pages. The next threshold must be at least 100 + (2 * 64), or 228 pages.

```
select @@thresh_hysteresis

-----------
         64
```

```
sp_addthreshold mydb, user_log_dev, 228,
    sp_thresholdaction
```

Creating threshold procedures

Sybase does not supply threshold procedures. You must create these procedures yourself to ensure that they are tailored to your site's needs.

Suggested actions for a threshold procedure include writing to the server's error log and dumping the transaction log to increase the amount of log space. You can also execute remote procedure calls to an Open Server or to XP Server. For example, if you include the following command in sp_thresholdaction, it executes the procedure mail_me on an Open Server:

```
exec openserv...mail_me @dbname, @segment
```

See Chapter 16, "Using Extended Stored Procedures," in the *Transact-SQL User's Guide* for more information on using extended stored procedures and XP Server.

This section provides some guidelines for writing threshold procedures, as well as two sample procedures.

Declaring procedure parameters

Adaptive Server passes four parameters to a threshold procedure:

- *@dbname*, varchar(30), which contains the database name
- *@segmentname*, varchar(30), which contains the segment name
- *@space_left*, int, which contains the space-left value for the threshold
- *@status*, int, which has a value of 1 for last-chance thresholds and 0 for other thresholds

These parameters are passed by position rather than by name. Your procedure can use other names for these parameters, but must declare them in the order shown and with the datatypes shown.

Generating error log messages

You should include a print statement near the beginning of your procedure to record the database name, segment name, and threshold size in the error log. If your procedure does not contain a print or raiserror statement, the error log will not contain any record of the threshold event.

The process that executes threshold procedures is an internal Adaptive Server process. It does not have an associated user terminal or network connection. If you test your threshold procedures by executing them directly (that is, using execute *procedure_name*) during a terminal session, you see the output from the print and raiserror messages on your screen. When the same procedures are executed by reaching a threshold, the messages go to the error log. The messages in the log include the date and time.

For example, if sp_thresholdaction includes this statement:

```
print "LOG DUMP: log for '%1!' dumped", @dbname
```

Adaptive Server writes this message to the error log:

```
00: 92/09/04 15:44:23.04 server: background task message: LOG DUMP: log for
'pubs2' dumped
```

Dumping the transaction log

If your sp_thresholdaction procedure includes a dump transaction command, Adaptive Server dumps the log to the devices named in the procedure. dump transaction truncates the transaction log by removing all pages from the beginning of the log, up to the page just before the page that contains an uncommitted transaction record.

When there is enough log space, suspended transactions are awakened. If you abort transactions rather than suspending them, users must resubmit them.

Generally, dumping to a disk is *not* recommended, especially to a disk that is on the same machine or the same disk controller as the database disk. However, since threshold-initiated dumps can take place at any time, you may want to dump to disk and then copy the resulting files to offline media. (You will have to copy the files back to the disk to reload them.)

Your choice will depend on:

- Whether you have a dedicated dump device online, loaded and ready to receive dumped data

- Whether you have operators available to mount tape volumes during the times when your database is available
- The size of your transaction log
- Your transaction rate
- Your regular schedule for dumping databases and transaction logs
- Available disk space
- Other site-specific dump resources and constraints

A simple threshold procedure

Following is a simple procedure that dumps the transaction log and prints a message to the error log. Because this procedure uses a variable (*@dbname*) for the database name, it can be used for all databases in Adaptive Server:

```
create procedure sp_thresholdaction
    @dbname varchar(30),
    @segmentname varchar(30),
    @free_space int,
    @status int
as
dump transaction @dbname
    to tapedump1
print "LOG DUMP: ’%1!’ for ’%2!’ dumped",
        @segmentname, @dbname
```

A more complex procedure

The following threshold procedure performs different actions, depending on the value of the parameters passed to it. Its conditional logic allows it to be used with both log and data segments.

The procedure:

- Prints a “LOG FULL” message if the procedure was called as the result of reaching the log’s last-chance threshold. The status bit is 1 for the last-chance threshold and 0 for all other thresholds. The test if (@status&1) = 1 returns a value of “true” only for the last-chance threshold.
- Verifies that the segment name provided is the log segment. The segment ID for the log segment is always 2, even if the name has been changed.

- Prints “before” and “after” size information on the transaction log. If the log did not shrink significantly, a long-running transaction may be causing the log to fill.
- Prints the time the transaction log dump started and stopped, helping gather data about dump durations.
- Prints a message in the error log if the threshold is not on the log segment. The message gives the database name, the segment name and the threshold size, letting you know that the data segment of a database is filling up.

```
create procedure sp_thresholdaction
     @dbname          varchar(30),
     @segmentname     varchar(30),
     @space_left      int,
     @status          int
as
declare @devname varchar(100),
           @before_size int,
           @after_size int,
           @before_time datetime,
           @after_time datetime,
           @error int

/*
** if this is a last-chance threshold, print a LOG FULL msg
** @status is 1 for last-chance thresholds,0 for all others
*/
if (@status&1) = 1
begin
      print "LOG FULL: database ’%1!’", @dbname
end

/*
** if the segment is the logsegment, dump the log
** log segment is always "2" in syssegments
*/
if @segmentname = (select name from syssegments
               where segment = 2)
begin
     /* get the time and log size
     ** just before the dump starts
     */
     select  @before_time = getdate(),
        @before_size = reserved_pgs(id, doampg)
      from sysindexes
      where sysindexes.name = "syslogs"
```

```
        print "LOG DUMP: database '%1!', threshold '%2!'",
          @dbname, @space_left

        select @devname = "/backup/" + @dbname + "_" +
          convert(char(8), getdate(),4) + "_" +
          convert(char(8), getdate(), 8)

        dump transaction @dbname to @devname

        /* error checking */
        select @error = @@error
        if @error != 0
        begin
             print "LOG DUMP ERROR: %1!", @error
        end

        /* get size of log and time after dump */
        select @after_time = getdate(),
             @after_size = reserved_pgs(id, doampg)
             from sysindexes
             where sysindexes.name = "syslogs"

        /* print messages to error log */
        print "LOG DUMPED TO: device '%1!", @devname
        print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
             @before_size, @after_size
        print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time
end          /* end of 'if segment = 2' section */
else         /* this is a data segment, print a message */
begin
        print "THRESHOLD WARNING: database '%1!', segment '%2!' at '%3!'
pages", @dbname, @segmentname, @space_left
end
```

Deciding where to put a threshold procedure

Although you can create a separate procedure to dump the transaction log for each threshold, it is easier to create a single threshold procedure that is executed by all log segment thresholds. When the amount of free space on a segment falls below a threshold, Adaptive Server reads the systhresholds table in the affected database for the name of the associated stored procedure, which can be any of:

- A remote procedure call to an Open Server
- A procedure name qualified by a database name (for example, sybsystemprocs.dbo.sp_thresholdaction)
- An unqualified procedure name

If the procedure name does not include a database qualifier, Adaptive Server looks in the database where the shortage of space occurred. If it cannot find the procedure there, and if the procedure name begins with the characters "sp_", Adaptive Server looks for the procedure in the sybsystemprocs database and then in master database.

If Adaptive Server cannot find the threshold procedure, or cannot execute it, it prints a message in the error log.

Disabling free-space accounting for data segments

Use the no free space acctg option to sp_dboption, followed by the checkpoint command, to disable free-space accounting on non-log segments. You *cannot* disable free-space accounting on the log segment.

When you disable free-space accounting, only the thresholds on your log segment monitor space usage; threshold procedures on your data segments will not execute when these holds are crossed. Disabling free-space accounting speeds recovery time because free-space counts are not recomputed during recovery for any segment except the log segment.

The following example turns off free-space accounting for the production database:

```
sp_dboption production,
     "no free space acctg", true
```

Warning! If you disable free-space accounting, system procedures cannot provide accurate information about space allocation.

Index