SYBASE®

New Functionality in Adaptive Server 12.5.2

Adaptive Server Enterprise

Version 12.5.2

DOCUMENT ID: DC35401-01-1252-01

LAST REVISED: April 2004

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

Audience

This manual is for Sybase® System Administrators and Database Owners who are using Adaptive Server® version 12.5.2. It discusses the new features included in this versions of Adaptive Server.

Adaptive Server version 12.5.2 is an overlay release, which means you install new binaries over your current version of Adaptive Server. See the cover letter for more information.

How to use this book

This manual includes the following:

- Chapter 1, “The Statement Cache” – describes the statement cache, which is used for saving cached SQL code.
- Chapter 2, “XML Services” – describes changes and extensions to the XML Services feature.
- Chapter 3, “Real Time Messaging Services” – describes the Real Time Messaging Services option.
- Chapter 4, “Web Services Consumer” – describes the Web Services option.
- Chapter 5, “IPv6 Support” – describes the implementation of internet protocol version 6 (IPv6).
- Chapter 6, “Changes to Enhanced Full-Text Search Capabilities” – describes new features in Enhanced Full-Text Search (EFTS).
- Chapter 7, “Security Changes” – describes the new security features.
- Chapter 8, “Making compressed database dumps” – describes the compression parameter of the dump command, which allows you to compress your database dumps.
- Chapter 9, “Making password-protected database dumps” – describes the password parameter of the dump and load database commands, which allows you to password-protect your database dumps.
- Chapter 10, “Configuring Linux Adaptive Server for Failover on Veritas 2.1” – describes how to configure Adaptive Server for failover on Veritas Cluster Server.

- Chapter 11, “Large Memory Support for 32-Bit Linux” – describes large memory support for 32-bit Red Hat Enterprise Linux 3.0 (RHEL 3) and Red Hat Advanced Server Linux 2.1.
- Chapter 12, “Changes to Global Variables, Commands, and Stored Procedures” – describes new and changed global variables, functions, commands, and stored procedures.

Related documents

The Sybase Adaptive Server Enterprise documentation set comprises these documents:

- 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 Administration Guide* – provides in-depth information about administering servers and databases. This manual includes instructions and guidelines for managing physical resources, security, user and system databases, and specifying character conversion, international language, and sort order settings.
- *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.

❖ **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.

❖ **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 maintenance

❖ **Finding the latest information on EBFs and software maintenance**

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

2 Select EBFs/Maintenance. 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/Maintenance report, or click the product description to download the software.

Conventions

In the regular text of this document, the names of files and directories appear in *italics*, for example:

- In Windows NT: *%SYBASE%\bin*
- In UNIX platforms: *$SYBASE*

Note Substitute your Sybase installation drive and directory for *$SYBASE* in UNIX, and *%SYBASE%* in Windows NT.

Table 1 details the typographic (font and syntax) conventions as used in this document.

Table 1: Font and syntax conventions for this document

Element	Example
Command names, command option names, database names, datatypes, utility names, utility flags, and other keywords ar in e Helvetica.	dsedit
Variables, or words that stand for values that you fill in, are in *italics*.	`select` *column_name* `from` *table_name* `where` *search_conditions*
Parentheses must be typed as part of the command.	`compute row_aggregate (column_name)`

Element	Example
Curly braces indicate that at least one of the enclosed options is required by the command (see comma).	`{cheese, sauce}` **Note** Do not type the curly braces.
Brackets mean that choosing one or more of the enclosed options is optional.	`[anchovies, pineapple, bell_peppers]` **Note** Do not type the brackets.
The *vertical bar* means you may select only one of the options shown.	`{cash | check | credit}` **Note** Do not type the curly braces.
The *comma* means you may choose as many of the options shown as you like; be sure to separate multiple choices in a command with commas.	`[extra_cheese, avocados, sour_cream]` **Note** Do not type the brackets.
An *ellipsis* (...) means that you can *repeat* the unit that the ellipsis follows as many times as you like.	`buy` *thing* `= price [cash | check | credit] [,` *thing* `= price [cash | check | credit] ]...` • You must buy at least one *thing* (item) and give its price. • You may choose a method of payment: one of the options enclosed in square brackets. • You may choose also to buy additional items: as many of them as you like. For each item you buy, provide its name, its price, and (optionally) a method of payment.
Syntax statements, which display the utility's syntax including all its options, appear as shown here, either in san serif font for flags and options (-v), or italics for user-supplied values (*username*).	charset [-P*password*] [-S*server*] [-I*interface*] *sort_order* | *charset*

Examples that illustrate computer output appear in `Courier`, as shown:	`pub_id pub_name city state`
	`------ --------- ------- -----`
	`0736 New Age Books Boston MA`
	`0877 Binnet & Hardley Washington DC`
	`(2 rows affected)`

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

The Statement Cache

This chapter describes the statement cache, which is used for saving SQL of cached statements. Adaptive Server compares incoming SQL to its cached SQL, and if they are equal, it executes the plan of the SQL already saved. This allows the application to amortize the costs of query compilation across several executions of the same statement.

Setting the statement cache

The statement cache allows Adaptive Server to store the text of ad hoc SQL statements. Adaptive Server compares a newly received ad hoc SQL statement to cached SQL statements and, if a match is found, uses the plan cached from the initial execution. In this way, Adaptive Server does not have to recompile SQL statements for which it already has a plan.

The statement cache is a server-wide resource, which allocates and consumes memory from the procedure cache memory pool. Set the size of the statement cache dynamically using the statement cache size configuration parameter. The syntax is:

```
sp_configure "statement cache size", size_of_cache
```

where *size_of_cache* is the size, in 2K pages. For example, to set your statement cache to 5000 2K pages, enter:

```
sp_configure "statement cache size", 5000
```

See “statement cache size configuration parameter” on page 8 for more information.

Consider the following when you configure memory for the statement cache:

- The amount of memory allocated for the procedure cache memory pool is the sum of the statement cache size and the procedure cache size configuration parameters. The statement cache memory is taken from the procedure cache memory pool. In the example above, the size of the procedure cache memory pool is increased by 5000 2K pages.
- statement cache size limits the amount of procedure cache memory by cached SQL text and plans. That is, Adaptive Server cannot use more memory for the statement cache than you have configured with the statement cache size configuration parameter.
- All procedure cache memory, including that memory allocated by the statement cache size configuration parameter, is available for stored procedures, which may replace cached statements on an LRU basis.
- Increase the max memory configuration parameter by the same amount configured for the statement cache. That is, if you have configured the statement cache size to be 100 2K pages, increase max memory by the same amount.
- If you have configured the statement cache with the statement cache size configuration parameter, you can disable and enable the statement cache at the session level with set statement cache. By default, the statement cache is on at the session level if it has been configured at the server level.
- Because each cached statement consumes one object descriptor, you must also increase the number of object descriptors accordingly, using the number of open databases configuration parameter. To estimate how many cached SQL statements to allow for, see “Statement cache sizing” on page 4.

Ad hoc query processing

Adaptive Server performs the following steps to process ad hoc SQL statements using the statement cache:

1 Adaptive Server parses the statement.

   If the statement should be cached (see “Caching conditions” on page 4), Adaptive Server computes a hash value from the statement. Adaptive Server uses this hash value to search for a matching statement in the statement cache (see “Statement matching criteria” on page 3).

- If a match is found in the statement cache, Adaptive Server skips to step 4.
- If a match is not found, Adaptive Server proceeds to step 2.

2 Adaptive Server caches the SQL statement text.

3 Adaptive Server wraps the SQL statement with a lightweight stored procedure and changes any local variables into procedure parameters. The internal representation of the lightweight procedure is not yet compiled into the plan.

4 Adaptive Server converts the SQL statement into an execute statement for the corresponding lightweight procedure.

- If there is no plan in the cache, Adaptive Server compiles the procedure and caches the plan. Adaptive Server compiles the plan using the assigned runtime values for the local variables.
- If the plan exists but is invalid, Adaptive Server returns to step 3 using the text of the cached SQL statement.

5 Adaptive Server then executes the procedure.

Statement matching criteria

Adaptive Server matches an ad hoc SQL statement to a cached statement by the SQL text and by login (particularly if both users have sa_role), user ID, database ID, and session state settings. The relevant session state consists of settings for the following set command parameters:

- forceplan
- jtc
- parallel_degree
- prefetch
- quoted_identifier
- sort_merge
- table count
- transaction isolation level
- chained (transaction mode)

Settings for these parameters determine the behavior of the plan Adaptive Server produces for a cached statement. For information on the set command and its parameters, see the *Adaptive Server Reference Manual*.

Note You must configure set chained on/off in its own batch if you enable the statement cache.

Caching conditions

- Adaptive Server currently caches select, update, delete, and insert select statements with at least one table reference.
- Statements are not cached if the abstract plan dump or abstract plan load parameters are enabled.
- Adaptive Server does not cache select into statements, cursor statements, dynamic statements, plain insert (not insert select) statements, and statements within stored procedures, views, and triggers. Statements that refer to temporary tables are not cached, nor are statements with language parameters transmitted as BLOB datatypes. Statements that are prohibitively large are not cached. Also, select statements that are part of a conditional if exists or if not exists clause are not cached.

Statement cache sizing

Each cached statement requires approximately 1K memory in the statement cache, depending on the length of the SQL text. Each cached plan requires at least 2K of memory in the procedure cache. To estimate the statement cache memory required, account for the following for each statement to be cached:

- The length of the SQL statement, in bytes, rounded up to the nearest multiple of 256.
- Approximately 100 bytes overhead.
- The size of the plan in the procedure cache. This size is equivalent to the size of a stored procedure plan containing only the cached statement. There may be duplicates of the plan for a single cached statement being used concurrently by two or more users.

Monitoring the statement cache

sp_sysmon reports on statement caching and stored procedure executions. The statement cache is monitored by the following counters:

- Statements Found in Cache – the number of times a query plan was reused. A low number of cache hits may indicate the statement cache is too small.
- Statements Not Found – indicates a lack of repeated SQL statements. The sum of statements found in cache and statements not found is the total number of eligible SQL statements submitted.
- Statements Cached – the number of SQL statements in cache. This is typically the same number as Statements Not Found. Smaller values for statements cached means the statement cache is full of active statements.
- Statements Dropped – the number of statements that were dropped instead of cached. A high value may indicate an insufficient amount of procedure cache memory or that you have configured a statement cache size that is too small.
- Statements Restored – the number of query plans regenerated from the SQL text. High values indicate an insufficient procedure cache size.
- Statements Not Cached – the number of statements Adaptive Server would have cached if the statement cache were enabled. However, Statements Not Cached does not indicate how many different statements were not cached.

This is sample output from sp_sysmon:

```
Procedure Cache Management        per sec      per xact       count  % of total
--------------------------  ------------  ------------  ----------  ----------
  Procedure Requests                 6.6           3.7          33         n/a
  Procedure Reads from Disk          1.0           0.6           5       15.2%
  Procedure Writes to Disk           0.4           0.2           2        6.1%
  Procedure Removals                 2.6           1.4          13         n/a
  Procedure Recompilations           0.8           0.4           4         n/a

  Recompilations Requests:
   Execution Phase                  0.6           0.3           3       75.0%
   Compilation Phase                0.2           0.1           1       25.0%
   Execute Cursor Execution         0.0           0.0           0        0.0%
   Redefinition Phase               0.0           0.0           0        0.0%

  Recompilations Reasons:
   Table Missing                    0.6           0.3           3         n/a
   Temporary Table Missing          0.2           0.1           1         n/a
   Schema Change                    0.0           0.0           0         n/a
```

```
   Index Change                    0.0          0.0          0        n/a
   Isolation Level Change          0.2          0.1          1        n/a
   Permissions Change              0.0          0.0          0        n/a
   Cursor Permissions Change       0.0          0.0          0        n/a

  SQLStatement Cache:
   Statements Cached               0.0          0.0          0        n/a
   Statements Found in Cache       0.7          0.0          2        n/a
   Statements Not Found            0.0          0.0          0        n/a
   Statements Dropped              0.0          0.0          0        n/a
   Statements Recompiled           0.3          0.0          1        n/a
   Statements Not Cached           1.3          0.0          4        n/a
```

Purging the statement cache

Run dbcc purgesqlcache to remove all the SQL statements from the statement cache. Any statements that are currently running are not removed.

See “Changes to dbcc” on page 12 for syntax information.

You must have the sa_role to run dbcc purgesqlcache

dbcc purgesqlcache prints the following message:

```
dbcc purgesqlcache
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator (SA)
role.
```

Printing statement summaries

Run dbcc prsqlcache to print summaries of the statements in the statement cache. The *oid* option allows you to specify the object ID of the statement to print, and the *printopt* option allows you to specify whether you print the trace description (specify 0) or the showplan option (specify 1). If you do not include any values for *oid* or *printopt*, dbcc prsqlcache displays the entire contents of the statement cache.

See “Changes to dbcc” on page 12 for syntax information.

You must have the sa_role to run dbcc prsqlcache

This provides information for all statements in the cache:

```
dbcc prsqlcache
```

```
Start of SSQL Hash Table at 0xfc67d830
Memory configured: 1000 2k pages            Memory used: 18 2k pages
Bucket# 625 address 0xfc67ebb8

SSQL_DESC 0xfc67f9c0
ssql_name *ss1248998166_0290284638ss*
ssql_hashkey 0x114d645e ssql_id 1248998166
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 8
SQL TEXT: select * from sysobjects where name like "sp%"

Bucket# 852 address 0xfc67f2d0
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0

End of SSQL Hash Table

DBCC execution completed. If DBCC printed error messages, contact a user with
```

Or you can get information about a specific object ID:

```
dbcc prsqlcache (1232998109, 0)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0

DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

This example specifies 1 in the printopt parameter for the showplan output:

```
dbcc prsqlcache (1232998109, 1)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
```

```
ssql_suid 1             ssql_uid 1       ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select  name from systypes where allownulls = 0

QUERY PLAN FOR STATEMENT 1 (at line 1).

      STEP 1
         The type of query is SELECT.
         FROM TABLE
            systypes
         Nested iteration.
         Table Scan.
         Forward scan.

         Positioning at start of table.
         Using I/O Size 2 Kbytes for data pages.
         With LRU Buffer Replacement Strategy for data pages.
         DBCC execution completed. If DBCC printed error messages,
         contact a user with
          System Administrator (SA) role.
```

statement cache size configuration parameter

statement cache size

Summary information	
Default value	0
Valid values	Size of cache in 2K pages
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator

The statement cache size parameter increases the server allocation of procedure cache memory and limits the amount of memory from the procedure cache pool used for cached statements. The statement cache feature is enabled server-wide:

```
statement cache size size_of_cache
```

Note You must configure set chained on/off in its own batch if you enable the statement cache.

Because cached statements are transformed into lightweight stored procedures, statement caching requires additional open object descriptors

Configuring memory for caches

Memory is the most important consideration when you are configuring Adaptive Server. Memory is consumed by various configuration parameters, the procedure cache, statement cache, and data caches. Setting the values of the various configuration parameters and the caches correctly is critical for good system performance.

The total memory allocated during system start-up is the sum of memory required for all the configuration needs of Adaptive Server. This value can be obtained from the read-only configuration parameter total logical memory. This value is calculated by Adaptive Server. The configuration parameter max memory must be greater than or equal to total logical memory. max memory indicates the amount of memory you will allow for Adaptive Server needs.

During server start-up, by default, Adaptive Server allocates memory based on the value of total logical memory. However, if the configuration parameter allocate max shared memory has been set, then the memory allocated will be based on the value of max memory. The configuration parameter allocate max shared memory enables a System Administrator to allocate the maximum memory that is allowed to be used by Adaptive Server, during server start-up.

The key points for memory configuration are:

- The System Administrator should determine the size of shared memory available to Adaptive Server and set max memory to this value.

- The configuration parameter allocate max shared memory can be turned on during start-up and runtime to allocate all the shared memory up to max memory with the least number of shared memory segments. A large number of shared memory segments has the disadvantage of some performance degradation on certain platforms. Check your operating system documentation to determine the optimal number of shared memory segments. Once a shared memory segment is allocated, it cannot be released until the server is restarted.
- The difference between max memory and total logical memory is additional memory available for the procedure and statement caches, data caches, or for other configuration parameters.

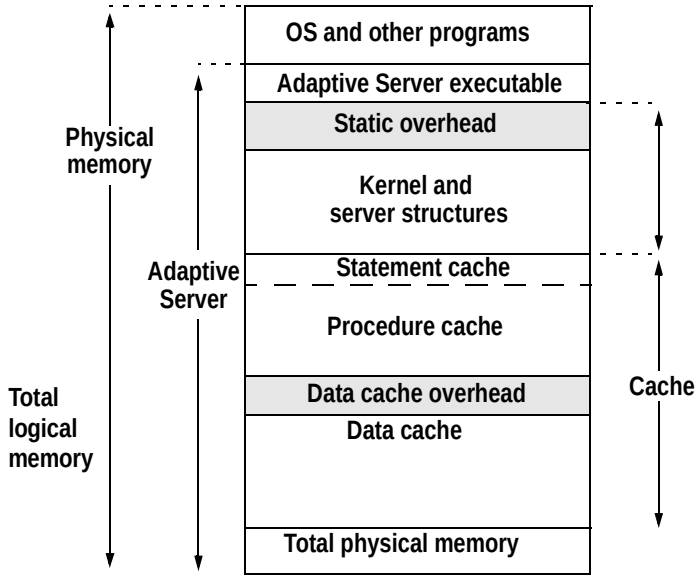
  The amount of memory to be allocated by Adaptive Server during start-up, is determined by either total logical memory or max memory. If this value too high:

  - Adaptive Server may not start if the physical resources on your machine are not sufficient.
  - If it does start, the operating system page fault rates may rise significantly and the operating system may need to be reconfigured to compensate.
  - Handling wider character literals requires Adaptive Server to allocate memory for string user data. Also, rather than statically allocating buffers of the maximum possible size, Adaptive Server allocates memory dynamically. That is, it allocates memory for local buffers as it needs it, always 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.
  - If you require Adaptive Server to handle more than 1000 columns from a single table, or process over 10000 arguments to stored procedures, the server must set up and allocate memory for various internal data structures for these objects. An increase in the number of small tasks that are performed repeatedly may cause performance degradation for queries that deal with larger numbers of such items. This performance hit increases as the number of columns and stored procedure arguments increases.
  - Memory that is allocated dynamically slightly degrades the server's performance.

- When Adaptive Server uses larger logical page sizes, all disk I/Os are performed in terms of the larger logical page sizes. For example, if Adaptive Server uses an 8K logical page size, it retrieves data from the disk in 8K blocks. This should result in an increased I/O throughput, although the amount of throughput is eventually limited by the controller's I/O bandwidth.

What remains after all other memory needs have been met is available for the procedure and statement cache, and the data cache. Figure 1-1 shows how memory is divided.

Figure 1-1: How Adaptive Server uses memory



Changes to commands

This section describes changes to commands for configuring and monitoring the statement cache.

Changes to dbcc

dbcc adds the prsqlcache and purgesqlcache parameters. The prsqlcache parameter allows you to print summaries of cached SQL statements, and the purgesqlcache parameter purges all statement cache entries, except those that are currently in use.

This is the partial syntax for dbcc:

```
dbcc prsqlcache[oid, printopt]
dbcc purgesqlcache
```

where:

- prsqlcache – prints summaries of cached SQL statements.
- *oid* – the object ID of the entry for which a summary is to be printed. Additional output is controlled by the value of the *printopt* variable. The value of *oid* can also be set to 0, in which case a brief summary is printed for every cached SQL statement, and the value of *printopt* is ignored.
- *printopt* – controls the output for prsqlcache when *oid* is set to a valid object ID. The *printopt* variable can be set to 0 or 1. When *printopt* is set to 1, showplan output is printed.
- purgesqlcache – results in all SQL statements being deleted from the statement cache except those currently in use.

See “Purging the statement cache” on page 6 for an example of dbcc prsqlcache and “Printing statement summaries” on page 6 for examples of dbcc purgesqlcache.

Changes to set

The statement cache adds the set statement cache parameter.

This is the partial syntax for set:

```
set statement_cache [on | off]
```

Parameter

statement_cache – enables or disables the statement cache at the session level. Once a use with SA priviledges configures the statement cache size to a non-0 value, the session can enable and disable the statement cache at the session level. If the statement cache is configured at the server level, by default it is enabled at the session level.

CHAPTER 2

XML Services

This chapter describes changes and extensions to the XML Services feature.

Overview

The changes in XML Services enhance three main areas:

- Extended XML query language for the xmlextract built-in function and xmltest predicate.
  - Support for XPath string functions:
    - toupper(...)
    - tolower(...)
    - normalize-space(...)
    - concat(...)
  - Support for parenthesized XPath expressions
  - XPath standards convergence:
    - Removal of nonstandard subscript expressions

    - Interpretation of the "//" operator
- Enhanced datatype support in the for xml clause of select statements:
    - text, image, binary, and varbinary
    - java.lang.String
- Enhanced datatype support for xmlextract, xmlparse, and xmlrepresentation:
    - xmlextract – can return the result XML fragment as java.lang.String.
    - xmlparse – return the parsed document as a varbinary datatype.
    - xmlrepresentation – can accept varbinary XML documents.
- Enhanced Java-XML sample code:
    - A Java-based ForXmlTree function to map SQL data to a hierarchic XML document
    - A Java-based OpenXml function to extract a SQL table from a hierarchic XML document
    - Java-based SQLX mapping functions for SQL queries that return multiple result sets

XML query language extensions

This section discusses extensions to the XML query language, specifically to XPath operators and functions. "XPath general syntax" on page 14, and "XPath string functions: general guidelines" on page 16, show general supported BNF syntax and general guidelines for XML function use.

XPath general syntax

XML Services supports the following XPath syntax; the extensions are in bold. Adaptive Server version 12.5.2 extends the syntax for *primary_expr* and adds syntax for *function_call*.

```
xpath::= or_expr
or_expr::= and_expr | and_expr TOKEN_OR or_expr
and_expr::= union_expr | union_expr TOKEN_AND and_expr
union_expr::= intersect_expr
```

```
    | intersect_expr TOKEN_UNION union_expr
intersect_expr::= comparison_expr
    | comparison_expr TOKEN_INTERSECT intersect_expr
comparison_expr::= range_exp
    | range_expr general_comp comparisonRightHandSide
general_comp::= TOKEN_EQUAL  | TOKEN_NOTEQUAL
    | TOKEN_LESSTHAN  | TOKEN_LESSTHANEQUAL
    | TOKEN_GREATERTHAN  | TOKEN_GREATERTHANEQUAL
range_expr::= unary_expr  | unary_expr TOKEN_TO unary_expr
unary_expr::= TOKEN_MINUS path_expr
    | TOKEN_PLUS path_expr
    | path_expr
comparisonRightHandSide::=  literal
path_expr::= relativepath_expr  | TOKEN_SLASH
    | TOKEN_SLASH   relativepath_expr
    | TOKEN_DOUBLESLASH   relativepath_expr
relativepath_expr::= step_expr
    | step_expr TOKEN_SLASH relativepath_expr
    | step_expr TOKEN_DOUBLESLASH relativepath_expr
step_expr::= forward_step predicates
    | primary_expr predicates
    | predicates
primary_expr::= literal | function_call | (xpath)
function_call::=
    tolower([xpath])
    | toupper([xpath])
    | normalize-space([xpath])
    | concat([xpath [,xpath]...])
forward_step::= abbreviated_forward_step
abbreviated_forward_step::= name_test
    | TOKEN_ATRATE name_test
    | TOKEN_PERIOD
name_test::= q_name  | wild_card | text test
text_test ::= TOKEN_TEXT TOKEN_LPAREN TOKEN_RPAREN
literal::= numeric_literal  | string_literal
wild_card::= TOKEN_ASTERISK
q_name::= TOKEN_ID
string_literal::= TOKEN_STRING
numeric_literal::= TOKEN_INT  | TOKEN_FLOATVAL|
    | TOKEN_MINUS TOKEN_INT
    | TOKEN_MINUSTOKEN_FLOATVAL
predicates::=
    | TOKEN_LSQUARE expr TOKEN_RSQUARE predicates
    | TOKEN_LSQUARE expr TOKEN_RSQUARE
```

XPath string functions: general guidelines

Adaptive Server version 12.5.2 extends XML query language for xmlextract and xmltest, supporting the following XPath string functions:

- toupper
- tolower
- normalize-space
- concat

This section describes general guidelines for using functions in XPath expressions. These guidelines apply to all the functions listed. All these examples use tolower, which returns a single argument in lowercase.

You can use a function call wherever you would use a step expression.

Example 1

Functions used as the top level of an XPath query are called top-level function calls. The following query shows tolower as a top-level function call:

```
select xmlextract
('tolower(//book[title]="Seven Years in Trenton"]//first-name)', text_doc)
from sample_docs where name_doc='bookstore'
----------------------------------------
joe
```

The parameters of a top-level function call must be an absolute path expression; that is, the parameter must begin with a slash (/) or a double slash (//).

Example 2

The parameters of a function call can be complex XPath expressions that include predicates. They can also be nested function calls:

```
select xmlextract
('//book[normalize-space(tolower(title))="seven years in trenton"]/author',
text_doc)
from sample_docs where name_doc='bookstore'

----------------------------------------
<author>
     <first-name>Joe</first-name>
     <last-name>Bob</last-name>
     <award>Trenton Literary Review
     Honorable Mention</award>
</author>
```

Example 3

You can use a function as a relative step, also called a relative function call. The following query shows tolower as a relative function call:

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'

-------------------------------------

joe
```

This example shows that the parameters of a relative function must be a relative path expression; that is, it cannot begin with a slash (/) or a double slash(//).

Example 4

Both top-level and relative functions can use literals as parameters. For example:

```
select xmlextract( 'tolower("aBcD")' ,text_doc),
    xmlextract( '/bookstore/book/tolower("aBcD")', text_doc)
from sample_docs where name_doc='bookstore'

 --------  ----------

 abcd      abcd
```

Example 5

String functions operate on the text of their parameters. This is an implicit application of text(). For example, this query returns a *first-name* element as an XML fragment:

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//firstname', text_doc)
from sample_docs where name_doc='bookstore'

------------------------------

<first-name>Joe</first-name>
```

The following query returns the text of that *first-name* XML fragment:

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//first-name/text()', text_doc)
from sample_docs where name_doc='bookstore'

-------------------------------

Joe
```

The next query applies tolower to the *first-name* element. This function operates implicitly on the text of the element:

```
select xmlextract
('//book[title="Seven Years in Trenton"] //tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'

---------------------------------------------

joe
```

This has the same effect as the next example, which explicitly passes the text of the XML element as the parameter:

```
select xmlextract
( '//book[title="Seven Years in Trenton"]//tolower(first-name/text())',
text_doc)
from sample_docs where name_doc='bookstore'

--------------------------------------

joe
```

Example 6

You apply a relative function call as a step in a path. Evaluating that path produces a sequence of XML nodes, and performs a relative function call for each node.The result is a sequence of the function call results. For example, this query produces a sequence of *first_name* nodes:

```
select xmlextract( '/bookstore/book/author/first-name', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------
<first-name>Joe</first-name><first-name>Mary</first-name>
<first-name>Toni</first-name>
```

The query below replaces the last step of the previous query with a call to toupper, producing a sequence of the results of both function calls.

```
select xmlextract('/bookstore/book/author/toupper(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
JOEMARYTONI
```

Now you can use concat to punctuate the sequence of the function results. See the example in "concat" on page 21.

Example 7

tolower, toupper, and normalize-space each have a single parameter. If you omit the parameter when you specify these functions in a relative function call, the current node becomes the implicit parameter. For instance, this example shows a relative function call of tolower, explicitly specifying the parameter:

```
select xmlextract
('//book[title="Seven Years in Trenton"]//tolower(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------------------------
joe
```

This example of the same query specifies the parameter implicitly:

```
select xmlextract
('//book[title="Seven Years in Trenton"]//first-name/tolower()', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
joe
```

You can also specify parameters implicitly in relative function calls when the call applies to multiple nodes. For example:

```
select xmlextract('//book//first-name/tolower()', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
joemarymarytoni
```

Functions

This section describes the individual functions that enhance XML Services.

tolower and toupper

Description — tolower and toupper return their argument values in lowercase and uppercase, respectively.

Syntax

tolower(*string-parameter*)

toupper(*string-parameter*)

Example — This example uses toupper to return the argument value in uppercase.

```
select xmlextract
('//book[title="Seven Years in Trenton"]//toupper(first-name)', text_doc)
from sample_docs where name_doc='bookstore'
----------------------------------------
JOE
```

normalize-space

Description — Makes two changes when it returns its argument value:

- It removes leading and trailing white-space characters.
- It replaces all substrings of two or more white-space characters that are not leading characters with a single white-space character.

Syntax

normalize-space(*string-parameter*)

Examples — This example applies normalize-space to a parameter that includes leading and trailing spaces, and embedded newline and tab characters:

```
select xmlextract
('normalize-space(" Normalize  space   example. ")', text_doc)
from sample_docs where name_doc='bookstore'
------------------------
Normalize space example.
```

normalize-space and tolower or toupper are useful in XPath predicates, when you are testing values whose use of white space and case is not known. The following predicate is unaffected by the case and whitespace usage in the *title* elements:

```
select xmlextract
('//magazine[normalize-space(tolower(title)="tracking trenton")]//price',
text_doc)
from sample_docs where name_doc='bookstore'
-------------------------
<price>55</price>
```

concat

Description — concat returns the string concatenation of the argument values. It has zero or more parameters.

Syntax — concat(*string-parameter* [,*string-parameter*]...)

Examples — concat can return multiple elements in a single call of xmlextract. For example, the following query returns both *first-name* and *last-name* elements:

```
select xmlextract('//author/concat(first-name, last-name)', text_doc)
from sample_dcs where name_doc='bookstore'
---------------------------------------
JoeBobMaryBobToniBob
```

You can also use concat to format and punctuate results. For example:

```
select xmlextract
('//author/concat(",first(",first-name, ")-last(",last-name, ") ")' ,
text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
first(Joe)-last(Bob) first(Mary)-last(Bob) first(Toni)-last(Bob)
```

Parenthesized expressions

Adaptive Server 12.5.2 supports parenthesized expressions. “XML query language extensions” on page 14 describes the general syntax of parenthesized expressions in XPath. The following sections describe how to use parentheses with subscripts and unions.

Parentheses and subscripts

Subscripts apply to the expression that immediately precedes them. Use parentheses to group expressions in a path. The examples in this section illustrate the use of parentheses with subscripts.

The following general example, which does not use subscripts, returns all titles in the book element.

```
select xmlextract('/bookstore/book/title', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Treanton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

To list only the first title, you can use the "[1]" subscript, and enter this query:

```
select xmlextract
('/bookstore/book/title[1]', text_doc)
from sample_docs where name_doc='bookstore'
-----------------------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Treanton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

However, the above query does not return the first title in the bookstore. It returns the first title in each book. Similarly, the following query, which uses the "[2]" subscript, returns the second title of each book, not the second title in the bookstore. Because no book has more than one title, the result is empty.

```
select xmlextract
('/bookstore/book/title[2]', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------
NULL
```

These queries return the *i*th title in the book, rather than in the bookstore, because the subscript operation (and predicates in general) applies to the immediately preceding item. To return the second title in the overall bookstore, rather than in the book, use parentheses around the element to which the subscript applies. For example:

```
select smlextract
('(/bookstore/booktitle)[2]', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
<title>History of Trenton</title>
```

You can group any path with parentheses. For example:

```
select xmlextract('(//title)[2]', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------------
<title>History of Trenton</title>
```

Parentheses and unions

You can also use parentheses to group operations within a step. For example, the following query returns all book titles in the bookstore.

```
select xmlextract('/bookstore/book/title', text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Who's Who in Trenton</title>
```

The above query returns only book titles. To return magazine titles, change the query to:

```
select xmlextract('/bookstore/magazine/title' ,text_doc)
from sample_docs where name_doc='bookstore'
-------------------------------------
<title>Tracking Trenton</title>
```

To return the titles of all items in the bookstore, you could change the query as follows:

```
select xmlextract('/bookstore/*/title', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

If the bookstore contains elements other than books and magazines—such as calendars and newspapers—you can query only for book and magazine titles by using the union (vertical bar) operator, and parenthesizing it in the query path. For example:

```
select xmlextract('/bookstore/(book|magazine)/title', text_doc)
from sample_docs where name_doc='bookstore'
---------------------------------
<title>Seven Years in Trenton</title>
<title>History of Trenton</title>
<title>Tracking Trenton</title>
<title>Trenton Today, Trenton Tomorrow</title>
<title>Whos Who in Trenton</title>
```

Changes in support of the XPath standard

Adaptive Server 12.5.2 includes these changes to the XPath language supported by the xmlextract function and the xmltest predicate:

- Removal of nonstandard subscript expressions
- Changes in the interpretation of the "//" operator

These changes align xmlextract and xmltest with the XPath standard.

Removal of nonstandard subscript expressions

Adaptive Server 12.5.2 no longer supports two features supported in Adaptive Server 12.5.1:

- Subscript expressions with negative values. For example:

  ```
  "/bookstore/book[-2]
  ```

- Subscript ranges in which the first number is high than the second. For example:

  ```
  "/bookstore/book[5 to 3]
  ```

Changes in the interpretation of the "//" operator

The XPath "//" operator represents an arbitrary number of arbitrary steps. Thus, these two query expressions are equivalent:

```
/bookstore//first-name
```

```
/bookstore/(* | */* | */*/* | */*/*/* | */*/*/*/* )/first-name
```

In Adaptive Server 12.5.2, the interpretation of the "//" operator has changed to reflect the XPath standard.

Consider a predicate intended to return the title of each book whose author's first name is "Mary." You could write this query like this:

```
(a) /bookstore/book[author/first-name = "Mary"]/title
```

In Adaptive Server 12.5.1, you could also reference first-name in the predicate using the "//" operator as follows:

```
(b) /bookstore/book[//first-name = "Mary"]/title
```

Adaptive Server 12.5.1 interpreted the leading "//" operator as a relative reference to all first-name elements contained in the current book element.

The XPath standard, however, specifies that a leading "//" operator is an absolute reference that references every first-name element in the entire document. To reference the first-name elements contained in the current book, you must precede the "//" operator with the "." operator to indicate the current context:

```
(c) /bookstore/book[.//first-name = "Mary"]/title
```

Adaptive Server 12.5.2 adopts the XPath standard interpretation of the "//" operator. Queries that were written in form (b) with Adaptive Server 12.5.1 should be written in form (c) with Adaptive Server 12.5.2.

Queries written in form (b) with Adaptive Server 12.5.2 will raise an exception:

```
select xmlextract( '/bookstore/book[//first-name = "Mary"]/title', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------------------------
Msg 14833, Level 16, State 0:
Line 1: Absolute paths inside a filter operator are not supported.
```

You should write such a query in the ASE 12.5.2 form, that is, form (c):

```
select xmlextract( '/bookstore/book[.//first-name ="Mary"]/title', text_doc)
from sample_docs where name_doc='bookstore'
--------------------------------------------------
<title>History of Trenton</title>
```

Extended datatype support for *xmlextract*, *xmlparse*, and *xmlrepresentation*

Adaptive Server 12.5.2 extends datatype support for these functions:

- xmlextract
- xmlparse
- xmlrepresentation

Changes to *xmlextract*

The returns clause of xmlextract now supports java.lang.String.

```
returns_type ::= [,] returns { varchar[(integer)] | text | image |
    java.lang.String }
```

Changes to *xmlparse*

xmlparse now includes an optional returns clause that lets you specify the datatype of the returned parsed XML document.

```
xmlparse_call ::=
xmlparse(general_string_expression[options_parameter] [returns_type])

options_parameter ::= [,] option option_string

returns type ::= [,] returns {image | binary | varbinary[(integer)]}
```

If you omit the returns clause, the default is returns image.

Changes to *xmlrepresentation*

In addition to the image datatype, xmlrepresentation now supports binary and varbinary datatypes for the XML document operand. That is, xmlrepresentation can examine an *image*, *binary*, or *varbinary* parameter, and return an integer value indicating whether the parameter contains parsed XML data.

Extended datatype support in *for xml* mappings

The for xml clause of the SQL select statement maps the query's result set to a SQLX-formatted XML document. See "for xml clause" in Chapter 4, "XML Mapping Functions," in *XML Services* for syntax and usage information.

Adaptive Server 12.5.2 extends the datatypes that can be specified in a select...for xml statement to include:

- binary and varbinary
- image
- text
- java.lang.String

binary, varbinary, and image data are represented in the generated XML document as either hex or base64 values. You can specify the value type by using the option "binary = { hex | base64 }" as described in "SQLX option definitions" in Chapter 4 of *XML Services*.

Examples

These two examples show different versions of a SQLX result set containing the binary value 0x123abc.

This example specifies a hex value:

```
select 0x123abc for xml option 'binary=hex'
------------------------------
<resultset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <row>
                <C1>123abc</C1>
        </row></resultset>
```

This example specifies a base64 value:

```
select 0x123abc for xml option 'binary=base64'
--------------------------------------
<resultset xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
>       <row>
                <C1>Ejq8</C1>
        </row>
```

Using Java functions to map hierarchic XML documents and SQL data

Adaptive Server 12.5.2 supports two new client-oriented Java-based XML functions for mapping data between SQL tables or result sets and hierarchic XML documents. They are:

- ForXmlTree – maps a set of SQL tables or result sets to a tree-structured XML document.
- OpenXml – extracts repeating data from a tree-structured XML document to a SQL table.

The following sections provide sample data and an overview and examples of how you can use ForXmlTree and OpenXml. For a more detailed description, see *$SYBASE/$SYBASE_ASE/sample/XML/xml-util.{doc, pdf}*.

Sample data and its tree-structured XML representation

SQL data is stored in tables, using foreign-key and primary-key columns to provide the tree-structured relationships between tables. When such data is depicted in XML, the tree-structured relationships are commonly represented with nested elements.

For example, consider tables with the data shown in Table 2-1.

Table 2-1: Sample tables

Table data
depts(dept_id, dept_name)
emps(emp_id, emp_name, dept_id)
emp_phones(emp_id, phone_no
projects(project_id, dept_id)

The tree-structured XML representation of the data in Table 2-1 is:

```
<sample xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<depts>
<dept>
      <dept_id>D123</dept_id>
      <dept_name>Main</dept_name>
       <emps>
           <emp>
               <emp_id>E123</emp_id>
               <emp_name>Alex Allen</emp_name>
               <salary>912.34</salary>
```

```
            <phones>
                <phone><phone_no>510.555.1987</phone_no></phone>
                <phone><phone_no>510.555.1876</phone_no></phone>
                 <!-- other phone elements for this emp -->
            </phones>
        <!--  Other emp elements for this dept  -- >
    </emps>

    <projects>
         <project>
             <project_id>PABC</project_id>
             <budget>598.76</budget>
         </project>
         <!--  Other project elements for this dept - ->
     </projects>
 </dept>
<!--  other dept elements for this set of depts. -->
</depts>
</sample>
```

Using *ForXmlTree* to map SQL data to hierarchic XML

The new Java-based function ForXmlTree maps a set of SQL tables or result sets to a tree-structured XML document. It is based on the for xml clause of the SQL select command, which was introduced in Adaptive Server 12.5.1.

select...for xml performs these tasks:

- Maps a single SQL result set to a single XML document.
- Generates a direct mapping of the SQL result set to XML. For example, if select returns a result set with 1000 rows, each having 20 columns, then the XML document returned by for xml has 1000 elements for the rows, each having 20 elements for the columns.

The new Java-based function ForXmlTree:

- Can be invoked in the SQL server, a client command line, or a client or server Java application.
- Maps a collection of results sets to a single tree-structured XML document.
- Requires a *<forxmltree>* specification argument, which describes the desired output tree and the SQL data to be included at each node of the tree.

- Generates a for xml-style mapping of XML data at each node of the output tree-structured XML document.

As a result, you can regard the ForXmlTree capability can as a two-dimensional for xml mapping. For example, the following <*forxmltree*> input for ForXmlTree generates the XML document shown in "Sample data and its tree-structured XML representation" on page 28.

```
1)  <!-- A forxmltree spec for depts-emps-phones-projects, with aggregation -->
2)  <forxmltree treename="sample">
3)  <node> <!-- The node element for depts -->
4)    <query>  select * from depts order by dept_id  </query>
5)    <options> tablename=depts rowname=dept </options>
6)    <link  variablename="@dept_id" columnname="dept_id" type="char(11)"  />
7)    <node>  <!-- The node element for emps, under depts -->
8)        <query>
9)            select emp_id, emp_name, salary from emps e
10)            where e.dept_id = @dept_id order by emp_id
11)        </query>
12)        <options> tablename=emps rowname=emp   </options>
13)        <link variablename="@emp_id" columnname="emp_id"  type="char(6)"  />
14)           <node> <!-- The node element for phones, under emps -->
15)             <query>
16)               select phone_no from emp_phones ep where ep.emp_id = @emp_id
17)             </query>
18)             <options> tablename=phones rowname=phone  </options>
19)           </node>  <!-- End the node for phones -->
20)    </node>  <!-- End the node for emps --
21)    <node>  <!-- The node element for projects, under dept -->
22)        <query>
23)            select project_id, budget from projects p
24)            where p.dept_id = @dept_id order by project_id
25)        </query>
26)        <options> tablename=projects rowname=project  </options>
27)    </node>  <!-- End the node for projects -->
28) </node>  <!-- End the node for depts -->
29) </forxmltree>
```

Using *OpenXml* to map hierarchic XML to SQL

The ForXmlTree function described in "Using ForXmlTree to map SQL data to hierarchic XML" on page 29 maps a collection of SQL tables or result sets to a hierarchic XML document. The OpenXml function reverses this process, and extracts the data for a SQL table from an input XML document.

OpenXml is similar to the xmlextract function, introduced in Adaptive Server 12.5.1, which extracts a specified data value from a given XML document. xmlextract specifies an XML document and a single XPath query expression. It returns the result of applying the XPath query to the XML document.

The new Java-based OpenXml function:

- Can be invoked from either a client command line or a client Java application. It is not intended for use in the SQL server.
- Requires arguments that include the specified XML document and a set of options that specify the XPath query that extracts the desired output rows and the Xpath queries that extract the desired columns in each output row.

Thus, you can regard OpenXml as a two-dimensional xmlextract.

OpenXml performs either or both of these actions:

- Generates a SQL script to create and populate a SQL table with the extracted data.
- Executes that script to create the SQL tables with the extracted data.

The following examples assume that the XML document in "Sample data and its tree-structured XML representation" on page 28 is stored in *example-document.xml*.

Example 8

This example shows four client command line calls to extract the depts, emps, emp_phones, and projects tables from the XML document.

```
java jcs.xmlutil.OpenXml -i "file:example-document.xml"  \
     -r "file:depts.opt"  -o "depts.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  \
       -r "file:emps.opt"  -o "emps.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  \
       -r "file:emp-phones.opt"  -o "emp-phones.sql"

java jcs.xmlutil.OpenXml -i "file:example-document.xml"  \
       -r "file:projects.opt"  -o "projects.sql"
```

Example 9

This example shows the contents of the options that the command line calls in Example 8 reference. These options specify the data that the calls for OpenXml should extract, and the SQL table in which they should be stored.

```
-- Content of input file "depts.opt
"tablename='depts_ext'
rowpattern='//dept'
columns=
  '   dept_id char( 4 ) "/@dept_id"
      dept_name varchar(50) "/@dept_name" '

-- Content of input options file "emps.opt"
tablename='emps_ext'
rowpattern='//dept/emps/emp'
columns=
  '   emp_id char( 4 ) "/emp_id/text()"
      emp_name varchar(50) "/emp_name/text()"
       dept_id char(4) "/../../@dept_id"
       salary dec(7,2) "/salary/text()"

'-- Content of input options file "emp-phones.opt"
tablename='emp_phones_ext'
rowpattern='/sample/dept/emps/emp/phone'
columns=  '   emp_id char( 4 ) "/../emp_id/text()"
       phone_no varchar(20) "/@phone_no" '

--Content of input options file "projects.opt"
tablename='projects_ext'
rowpattern='//dept/projects/project'
columns=
  '   project_id char( 4 ) "/project_id/text()"
      dept_id char(4) "/../../@dept_id"
      budget dec(7,2) "/budget/text()" '
```

Example 10

This example shows the SQL script generated by the first OpenXml call. The script creates and populates a table with the extracted depts table data. Subsequent OpenXml calls, shown in Example 8, generate similar scripts for the emps, emp_phones, and projects data.

```
-- output file depts.sql

create table depts_ext
   (dept_id char( 4 ) null, dept_name varchar(50) null )

insert into depts_ext values('D123',   'Main')

insert into depts_ext values('D234',  'Auxiliary')

insert into depts_ext values('D345',  'Repair')
```

Java SQLX mappings for multiple result set queries

The select ... for xml statement and the Java-based SQLX mapping functions map a single SQL result set to a SQLX-formatted XML document. Adaptive Server 12.5.2 provides a new Java-based SQLX mapping function, forxmlmultiplej, that maps multiple result sets of a SQL query to an XML document.

forxmlmultiplej

Description

Maps result sets of a SQL query, that can contain multiple result sets, to an XML document.

Syntax

forxmlmultiplej_function ::=
forxmlmultiplej(*sql_query_expression*, *option_string*)

Options

See "forxmlj, forxmldtdg, forxmlschemaj, forxmlallj" in Chapter 4, "XML mapping functions," in *XML Services* for a description of *sql_query_expression* and *option_string*.

Usage

- *swl_query_expression* can return multiple result sets, and can contain SQL print commands.
- See "Multiple result sets" in *$SYBASE/$SYBASE_ASE/sample/XML/Using-SQLX-mappings.htm* for examples and a complete description of forxmlmultiplej.

CHAPTER 3

Real Time Messaging Services

Adaptive Server Enterprise 12.5.2 includes a messaging functionality with the Real Time Data Services (RTDS) option package. This option simplifies the development of application that interact with messaging systems and databases.

RTDS allows you to capture transactions (data changes) in an Adaptive Server database and deliver them as events to external applications in real time. These data changes, or events, are delivered to applications through a Java Messaging Service (JMS) message bus, provided by TIBCO Enterprise™ for JMS. You can send messages to – or retrieve messages from – the message provider using Transact-SQL, provided by Adaptive Server.

For a complete description of the Real Time Messaging Services option, see the *Real Time Data Service User's Guide*.

Sending and receiving messages from a queue

Using the msgsend and msgrecv functions, Transact-SQL applications can send messages to – or read messages from – a JMS queue. A message body, or payload, can be constructed with the application, or it can contain character or binary data from relational tables. You can construct the values of message properties from relational data or from the Adaptive Server application, and you can specify message properties when you send the message.

The Adaptive Server application can process the messages read from the JMS queue or it can be inserted into relational tables. When executing the read operation, you can also specify a message selector to filter specific messages. Message properties in read messages can be individually processed by the Adaptive Server application.

Publishing and consuming messages from a topic

Transact-SQL applications can publish messages to – or consume messages from – a JMS topic with the msgpublish and msgconsume functions. You must first register a subscription with sp_msgadmin 'register', which creates a name that msgpublish, msgconsume, msgsubscribe, and msgunsubscribe can reference. A subscription can be registered as either durable or non-durable:

- A durable subscription retains messages for the message consumer even when the message consumer application is not connected. The message provider, not Adaptive Server, retains the message
- A nondurable subscription retains messages only when consumer applications are connected to the message provider.

You can specify a message selector to control the messages that come in, ensuring that only messages of interest are read.

You can use msgsubscribe to inform the JMS provider to hold messages until the Adaptive Server application is ready to process them, and you can use msgunsubscribe to inform the JMS provider that the application is no longer interested in messages on this subscription. msgunsubscribe can also delete durable subscriptions from the JMS provider. Message properties in messages that have been read can be individually processed by the application.

Examples

These examples provide a brief preview of the Transact-SQL messaging interface.

Example 1 This example sends a message to a queue:

```
select msgsend('hello world',
'tibco_jms:tcp://my_jms_host:7222?queue=queue.sample 'MESSAGE PROPERTY'
'city=Detroit')
```

Example 2 This example reads a message from a queue, with and without a filter:

```
select msgrecv('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample')
select msgrecv
('tibco_jms:tcp://my_jms_host:7222?queue=queue.sample 'MESSAGE SELECTOR'
'city=Detroit')
```

Example 3 This example publishes a message to a topic:

```
sp_msgadmin register, subscription, sub1,
'tibco_jms:tcp://my_jms_host:7222?topic=topic.sample',
select msgpublish('hello world', 'sub1' 'MESSAGE PROPERTY' 'city=Boston')
```

Example 4

This example consumes a message from a topic:

```
select msgconsume('sub1')
```

Example 5

This example illustrates working with properties:

```
select msgconsume('sub1')
declare @pcount integer
declare @curr integer
declare @pname varchar(100)
select @curr=1
select @pcount = msgpropcount()
while(@curr<=@pcount)
begin
select @name=msgpropname(@curr)
select msgproptype(@pname)
select msgpropvalue(@pname)
select @curr=@curr+1
end
```

CHAPTER 4

Web Services Consumer

A Web service is a self-contained, modular application that can be accessed through a network connection. Using a Web Service, the user trades performance for increased interoperability enforced by adherence to the Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), HTTP, and Extensible Markup Language (XML) open standards.

Regardless of the programming language in which it has been implemented, a Web service can be accessed from many different platforms and operating systems, thus greatly enhancing the ability for diverse applications to share data. By using many discrete Web services, each handling a limited set of specific tasks, business enterprises can dynamically and incrementally integrate by exposing their existing software in a secure and controlled environment. By providing a standardized means to invoke remote applications, Web services reduce the amount of code required for infrastructure. By enabling users to extract implementation from exposed interfaces (WSDL), Web services provide the tools needed to build a service-oriented architecture (SOA).

Components of Adaptive Server Enterprise Web Services

Adaptive Server Enterprise Web Services consists of two components: a Web Services Producer and a Web Services Consumer. Both components run independently of Adaptive Server and are enabled by the same license, ASE_WEBSERVICES.

- The Web Services Producer component allows client applications to access SQL and stored procedures in Adaptive Server using SOAP.

  For example, if a user-written application needs to access Adaptive Server through a firewall, that application can use the Web Services Producer to access Transact-SQL over the Internet using HTTP/HTTPS and SOAP.

For more information on the Web Services Producer, see Chapter 2, "Understanding the Web Services Producer" in the *Web Services User's Guide*.

- The Web Services Consumer component is new for Adaptive Server 12.5.2. The Web Services Consumer component enables Adaptive Server to access the Web services of other applications by mapping these external Web services to an Adaptive Server proxy table at runtime.

  For example, a user may need to integrate data from a Web service and relational data in Adaptive Server. The Web Services Consumer provides a way to dynamically map a Web service to a proxy table and also to map Web service output to a result set in Adaptive Server. Using this functionality, the user can retrieve and manipulate Web service output using T-SQL in a stored procedure, trigger, or view.

  For more information on the Web Services Consumer, see Chapter 3, "Understanding the Web Services Consumer" in the *Web Services User's Guide*.

sp_webservices

The sp_webservices stored procedure creates and manages the proxy tables used in the Consumer component of Web Services. sp_webservices has the following options:

- add – creates a proxy table.
- list – lists the proxy tables mapped to a WSDL file.
- modify – modifies timeout, user name, and password settings.
- remove – removes proxy tables mapped to a WSDL file.
- help – displays usage information for sp_webservices.

For more information about sp_webservices, see Chapter 5, "Using Adaptive Server Enterprise Web Services," in the *Web Services User's Guide*.

CHAPTER 5

IPv6 Support

This chapter describes the implementation of IPv6 support.

Starting Adaptive Server IPv6-aware

To make Adaptive Server IPv6-aware, you must start Adaptive Server with trace flag 7841, which allows Adaptive Server to determine IPv6 availability, and makes Adaptive Server IPv6-aware.

Note IPv6 is available on Sun Solaris 32- and 64-bit platforms. For more information about how to setup and administrate IPv6-enabled networking on your platform, see your operating system documentation.

IPv6 terminology

IPv6 transition process

Installing and configuring IPv6 is transparent to users. Here is some terminology that may be helpful.

- IPv4-only node – a node that implements only IPv4. An IPv4-only node has only an IPv4 address in the name service database.
- IPv6-only node – a node that implements only IPv6. An IPv6-only node has only an IPv6 address in the name service database.

- Dual node – a node that implements both IPv4 and IPv6. It is expected that the nodes that are upgraded from IPv4-only are upgraded to dual nodes.
- IPv6-enabled node – a node that implements a dual node and has at least one IPv6 interface configured.

IPv6 addressing terminology

- Link-local address – an IPv6 address that is usable only over a single link.
- Site-local address – an IPv6 address that can be used within a single-site.
- Global address – an IPv6 address that can be used across the global Internet.

IPv6 application types

- IPv6-unaware – an application that cannot handle IPv6 addresses.
- IPv6-aware – an application that can communicate with nodes that do not have IPv4 addresses. In some cases, this might be transparent to the application, for instance when the API hides the content and format of the actual addresses.
- IPv6-enabled – an application that, in addition to being IPv6-aware, takes advantage of some IPv6 features.
- IPv6-required – an application that requires some IPv6 features and cannot operate over IPv4.

CHAPTER 6

Changes to Enhanced Full-Text Search Capabilities

This chapter describes changes to Enhanced Full-Text Search (EFTS) capabilities in Adaptive Server version 12.5.2.

Note EFTS shipping on HP Tru64 is the existing 12.5.1 version of EFTS.

Changes to installation

Adaptive Server version 12.5.2 implements several changes and enhancements to EFTS. Because of these changes the following issues must be kept in mind when upgrading to EFTS for Adaptive Server version 12.5.2:

- Collections created with earlier version of EFTS are not compatible with this version of EFTS.
- Previously existing collections must be dropped before upgrading to the latest version of EFTS.
- Collections must be recreated after the upgrade.

The 12.5.2 version of EFTS is installed into a directory named *EFTS-12_5_2* and does not overwrite the existing *EFTS-12_5* directory. Once the old collections are dropped, you must edit the *SYBASE.csh* and *SYBASE.sh* files to change the value of the SYBASE_FTS environment variable to point to the *EFTS-12_5_2* directory.

New style files directory

The directory containing the Verity style files used by EFTS has changed. EFTS no longer uses the style files contained in *$SYBASE/$SYBASE_FTS/verity/common/style*.

Instead, EFTS now uses the style files contained in *$SYBASE/$SYBASE_FTS/verity/common/styles/txtsvr*.

New features

Adaptive Server version 12.5.2 introduces many new EFTS features.

Permissions for shutdown

Only users with sa_role can shut down EFTS.

New pseudo column *total_docs*

Starting with Adaptive Server version 12.5.2, EFTS returns an integer value for the total number of documents that match a search criteria. This value is returned in a new pseudo column called total_docs.

This is also useful when using the max_docs column, which limits the number of results returned.

index_any clauses support up to 16000 bytes

In earlier versions of Adaptive Server, EFTS did not support index_any clauses greater than 255 bytes. In Adaptive Server version 12.5.2, EFTS supports index_any clauses up to 16000 bytes.

The column definition for the proxy table is not altered, because the definition does not affect the size of the clause that is sent to EFTS.

Primary keys

In Adaptive Server version 12.5.2 and later, primary keys can be used as the text id column. You can create text indexes on tables that contain an identity column or a suitable primary key. Primary keys defined on a single decimal, numeric, int, smallint, or tinyint column are also eligible. Decimal and numeric columns must have a scale of zero

When creating a text index, EFTS first looks for an IDENTITY column in the source table to use as the ID column of the text index. If an IDENTITY column is not found, EFTS looks for a suitable primary key that it can use as the ID column of the text index.

CHAPTER 7

Security Changes

This chapter describes the security changes that are implemented in Adaptive Server version 12.5.2.

Introduction

Security features in Adaptive Server are grouped into the following four categories:

- Identification and authentication (I&A)
- Access control
- Accountability
- Encryption technologies

Changes have been made in Adaptive Server 12.5.2 to each of these four categories.

Identification and authentication

I&A refers to features used by Adaptive Server to positively identify a user. Once a user has been identified, access control mechanisms and individual accountability is enforced.

Adaptive Server 12.5.2 supports the following new and enhanced I&A features:

- Enhanced Kerberos
- LDAP user authentication
- PAM user authentication
- Enhanced login controls

Kerberos

Kerberos is a network authentication protocol that uses secret key cryptography so that a client can prove its identity to a server across a network connection. User credentials are obtained when the user logs in to the operating system, or by executing an authentication program. These credentials are then used by each application to perform authentication. Users only have to log in once, instead of having to log in to each application.

Adaptive Server 12.5.2 supports Kerberos through:

- CyberSafe Kerberos libraries on the following platforms:
  - Sun Solaris 32-bit
  - Sun Solaris 64-bit (new to Adaptive Server version 12.5.2)
  - Windows
  - AIX 32-bit
- MIT Kerberos libraries version 1.3.1 on the following platforms (new to Adaptive Server version 12.5.2):
  - Sun Solaris 32-bit
  - Sun Solaris 64-bit
  - Linux 32-bit
- Native libraries on the following platforms (new to Adaptive Server version 12.5.2):
  - Sun Solaris 32-bit
  - Sun Solaris 64-bit

- Linux 32-bit

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

Kerberos compatibility

Table 7-1 shows which variation of Kerberos is supported on which platforms.

Table 7-1: Adaptive Server version 12.5.2 Kerberos Interoperability

Hardware platforms	KDC server	GSS client
Solaris 32	CSF, AD, MIT	CSF, MIT, Native
Solaris 64	CSF, AD, MIT	CSF, MIT, Native
Linux 32	CSF, AD, MIT	MIT, Native
Windows 32	CSF, AD	CSF
AIX 32	CSF	CSF

Use the following keys to read the interoperability matrix:

- CSF – CyberSafe Ltd.
- AD – Microsoft Active Directory
- MIT – MIT version 1.3.1

Configuring Kerberos

The configuration process is similar, regardless of which variety of Kerberos is used. To configure Kerberos:

1 Set up Kerberos third-party software and create a Kerberos administrative user. To do this, you must:

   - Install Kerberos client software on machines where Open Client Server clients or Adaptive Server will run. The following client packages have been have been verified to be working:
     - CyberSafe TrustBroker 4.0
     - MIT Kerberos version 1.3.1

- Install the Kerberos KDC server on a separate, dedicated machine.

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

- Create an administrator account on the Kerberos server with administration privileges. This account is used for subsequent client actions such as creating principals from the client machines.

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

2 Add Kerberos principal for Adaptive Server *ase120srv* or *ase120srv@MYREALM*.

3 Extract the *keytab* file for principal *ase120srv@MYREALM* and store it as a file:

```
/krb5/v5srvtab
```

The following UNIX examples use the command line tool kadmin, available with CyberSafe or MIT Kerberos.There are also GUI tools available to aid in administration of Kerberos and users:

```
CyberSafe Kadmin:
% kadmin aseadmin
Principal - aseadmin@MYREALM
Enter password:
Connected to csfA5v01 in realm ASE.
Command: add ase120srv
Enter password:
Re-enter password for verification:
Principal added.
Command: ext -n ase120srv
Service Key Table File Name (/krb5/v5srvtab):
Key extracted.
Command: quit
Disconnected.
```

In a production environment, you must control the access to the *keytab* file. If a user can read the *keytab* file, he or she can create a server that impersonates your server.

Use chmod and chgrp so that */krb5/v5srvtab* is:

```
-rw-r----- 1 root sybase 45 Feb 27 15:42 /krb5/v5srvtab
```

When using Active Directory as the KDC, log in to the Domain Controller to add users and Adaptive Server principals. Use the Active Directory Users and Computers wizard to guide you through the creation of users and principals.

Extracting the *keytab* file for use with Adaptive Server requires an optional tool called ktpass, which is included in the Microsoft Support Tools package.

With Active Directory, extracting the *keytab* with ktpass is done as a separate step from creating the principal. The *keytab* file on Windows for Adaptive Server is located with the CyberSafe program files. For example, *c:\Program Files\CyberSafe\v5srvtab* is the expected location of Adaptive Server's *keytab* file when CyberSafe software is installed on the C: drive.

4 Add a Kerberos principal for the user "sybuser1" as "sybuser1@MYREALM".

5 Start Adaptive Server and use isql to log in as "sa". The following steps configure Adaptive Server parameters to use Kerberos security services, and create the user login account. These are the same on both Windows or UNIX machines:

- Change configuration parameter use security services to 1:

```
1> sp_configure 'use security services', 1
```

- Add new login for user, "sybuser1" and then add the user:

```
1> sp_addlogin sybuser1, password
```

6 Shut down Adaptive Server and modify administrative files and connectivity configuration files.

- On UNIX platforms, the *interfaces* file is under *$SYBASE/* and has an entry that looks similar to:

```
ase120srv
      master tli tcp myhost 2524
      query tli tcp myhost 2524
      secmech 1 .3.6.1.4.1.897.4.6.6
```

On Windows platforms, the *sql.ini* file is in *$SYBASE/ini*, and has an equivalent server entry that looks like:

```
[ase120srv]
master=TCP,myhost,2524
query=TCP,myhost,2524
secmech=1.3.6.1.4.1.897.4.6.6
```

- The *libtcl.cfg* or *libtcl64.cfg* file is located in *$SYBASE/$SYBASE_OCS/config/* on UNIX platforms.The SECURITY section should have an entry that looks similar to the following for CyberSafe Kerberos client libraries:

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

  A 64-bit CyberSafe Kerberos client library entry follows:

  ```
  [SECURITY]
  csfkrb5=libskrb64.so secbase=@MYREALM libgss= \
  /krb5/appsec-rt/lib/64/libgss.so
  ```

  For a machine that uses MIT Kerberos client libraries, the entry looks something like:

  ```
  [SECURITY]
  csfkrb5=libskrb.so secbase=@MYREALM
  libgss=/opt/mitkrb5/lib/libgssapi_krb5.so
  ```

  For a machine that uses Native OS provided libraries, such as Linux, it looks similar to:

  ```
  [SECURITY]
  csfkrb5=libskrb.so secbase=@MYREALM
  libgss=/usr/kerberos/lib/libgssapi_krb5.so
  ```

  On Windows NT, the *%SYBASE%\%SYBASE_OCS%\ini\libtcl.cfg* file contains an entry like:

  ```
  [SECURITY]
  csfkrb5=libskrb secbase=@MYREALM
  libgss=C:\WinNT\System32\gssapi32.dll
  ```

  Note Note the `libgss=<gss shared object path>` that specifies the GSS API library to be used. This is new in Adaptive Server 12.5.2 and Open Client Server 12.5.1. It is important that you distinctly locate the Kerberos Client libraries being used, especially when multiple versions are installed on a machine.

- Also check the *objectid.dat* under *$SYBASE/$SYBASE_OCS/config/* and make sure the *[secmech]* section has an entry for *csfkrb5*:

  ```
  [secmech]
  1.3.6.1.4.1.897.4.6.6 = csfkrb5
  ```

7 You can use environment variables to override default locations of *keytab* files, Kerberos configuration, and realm configuration files. This is Kerberos-specific behavior and may not work consistently on all platforms.

For example, the CSFC5KTNAME environment variable can be used on CyberSafe UNIX platforms to specify the *keytab* file:

```
% setenv CSFC5KTNAME /krb5/v5srvtab
```

For MIT Kerberos, the equivalent environment variable is KRB5_KTNAME.

See the vendor documentation for information about these environment variables.

Your application may also need to modify the environment variable for dynamic library search paths. On UNIX, the most commonly used environment variable is LD_LIBRARY_PATH; on Windows, PATH is typically set to include DLL locations. You may need to modify these environment variables to enable applications to load the third-party objects correctly. For example this command adds the location of CyberSafe 32-bit *libgss.so* shared object to the search path in a C-Shell environment:

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

8 Restart Adaptive Server. You should see the following log message during start-up:

```
00:00000:00000:2001/07/25 11:43:09.91 server
Successfully initialized the security mechanism
'csfkrb5'. The SQL Server will support use of this
security mechanism.
```

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

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

You can also use the encryption option:

```
$SYBASE/$SYBASE_OCS/bin/isql -Sase120srv –Vc
```

LDAP user authentication

LDAP externalizes authentication. When you are using LDAP, authentication decisions are based on whether Adaptive Server can successfully bind to a specified LDAP server on behalf of the user. To bind to an LDAP server, Adaptive Server uses a distinguished name (DN) extracted from the specified LDAP URL.

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

As of Adaptive Server version 12.5.2, LDAP-authenticated users must already exist as valid logins in Adaptive Server. To create new Adaptive Server logins for LDAP-authenticated users automatically, issue:

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

Alternatively, LDAP-authenticated users can be mapped to existing Adaptive Server users. For example:

```
sp_maplogin NULL, "externuser", "aseuser"
```

For more information, see “Mapping logins using sp_maplogin” on page 61.

Alternative algorithm for finding a user’s DN

Adaptive Server version 12.5.2 adds the following alternative algorithm to find a user’s DN:

- Bind to LDAP using a generic login
- Search the LDAP directory using a search string containing a user’s login ID
- Extract a DN from the first object returned by the search

Adaptive Server then uses this DN plus the password from the login packet to bind to LDAP.

New options added to *sp_ldapadmin*

The following new options have been added to sp_ldapadmin to support the new DN search algorithm:

- To allow users to specify an account that Adaptive Server can use for administrative searches, use:

```
sp_ldapadmin set_access_acct,
account_distinguished_name, account_password
```

The maximum length of *account_distinguished_name* is 255 characters. The maximum length of *account_password* is 64 characters and is encrypted using 3DES when stored on disk.

- To specify an alternative authentication algorithm, use:

```
sp_ldapadmin set_dn_lookup_url,
            <LDAP URL string for distinguished name lookup>
```

When set_dn_lookup_url is set to a non-NULL value, the alternative algorithm is used to authenticate the login with an LDAP Directory Server.

The URL string has a maximum length of 255 characters and is used to search for a distinguished name associated with the login name. The attribute name in the LDAP URL string should be set to obtain a DN, and the default is `entrydn`.

When more than one object is returned, only the first object is used. Authentication for a user login is done by binding the DN to the Directory Server specified by the primary or secondary LDAP URL. Set the LDAP URLs by issuing set_primary_url and set_secondary_url.

For example:

```
sp_ldapadmin set_access_acct, "cn=admin,
ou=People, dc=mycompany, dc=com", "admin_password"

sp_ldapadmin set_dn_lookup_url,
"ldap://myhost:398/ou=People,dc=mycompany,
dc=com?entrydn?sub?uid=*"

sp_ldapadmin set_primary_url, "ldap://myhost:389/"
```

For a complete description of sp_ldapadmin, see Chapter 12, “Changes to Global Variables, Commands, and Stored Procedures.”

This example uses the default Microsoft Active Directory schema found on Windows 2000 controllers.

```
1> sp_ldapadmin set_access_acct, 'cn=aseadmin,
cn=Users, dc=mycompany, dc=com', aseadmin secret
password
2> go
1> sp_ldapadmin set_dn_lookup_url,
'ldap://mydomainhostname:389/cn=Users,dc=mycompany,
dc=com?distinguishedName?sub?samaccountname=*'
2> go
```

```
1> sp_ldapadmin
set_primary_url,ldap://mydomainhostname:389/
2> go
```

The “aseadmin” user name has been added to the Active Directory server and granted read access to the trees and objects where users are found. The LDAP attribute specified by “distinguishedName” is obtained and used to authenticate the user. The filter specifies a search on attribute “samaccountname=*”; the * wildcard is replaced with the name from the Adaptive Server login record.

For example, “samaccountname=jqpublic” returns DN attribute “distinguishedName” with value “cn=John Q. Public, cn=Users,dc=mycompany, dc=com” to Adaptive Server. Adaptive Server uses this string to bind to ldap://mydomainhostname:389. If the bind is successful, authentication succeeds.

Pluggable Authentication Module (PAM) support

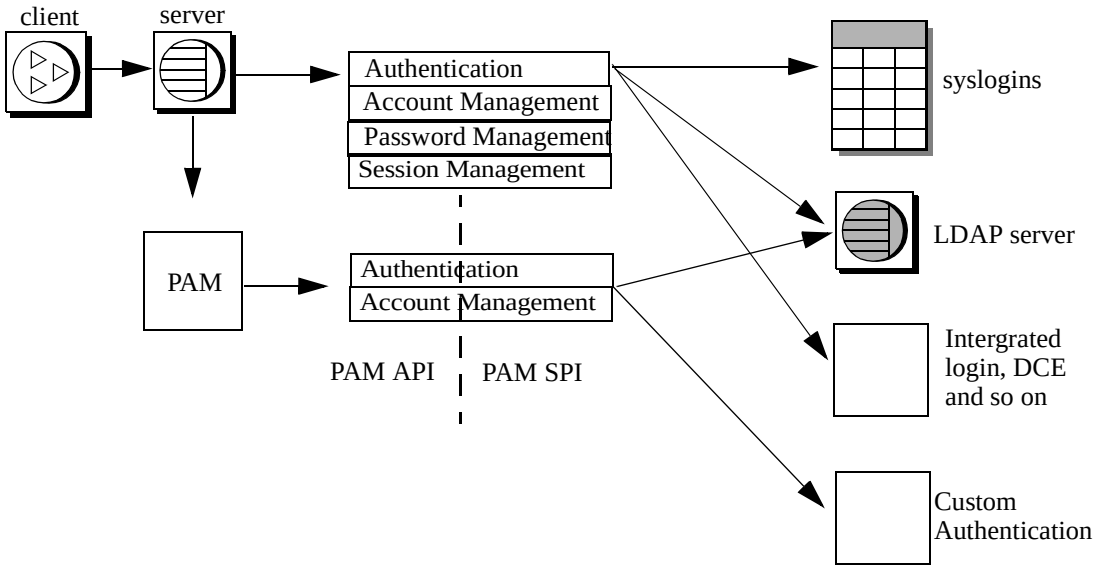
Adaptive Server version 12.5.2 introduces Pluggable Authentication Modules (PAM) support, which allows multiple authentication service modules to be stacked and made available without modifying the applications that require the authentication.

PAM integrates Adaptive Server more closely with Sun and Linux operating systems and simplifies the management and administration of user accounts and authentication mechanisms. PAM reduces the total cost of ownership through this closer integration. An additional benefit is that users can customize or write their own authentication and authorization modules.

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

Figure 7-1 shows how PAM works:

Figure 7-1: PAM architecture



Adaptive Server passes the login name and credentials obtained from the login packet to the PAM API. PAM loads a service provider module as specified in the Operating System configuration files and calls appropriate functions to complete the authentication process.

Enabling PAM in Adaptive Server

As of Adaptive Server version 12.5.2, PAM authenticated users must already exist as valid logins in Adaptive Server. To create new Adaptive Server logins for PAM-authenticated users automatically, issue:

```
sp_maplogin, PAM, NULL, "create login"
```

Alternatively, PAM-authenticated users can be mapped to existing Adaptive Server users. For example:

```
sp_maplogin NULL, "externuser", "aseuser"
```

For more information, see “Mapping logins using sp_maplogin” on page 61.

Determining which PAM module to use

Both Linux and Solaris have predefined PAM modules. You can choose to either use one of these modules or to create one of your own. When creating your own modules, follow the guidelines in your operating system documentation on creating a PAM module.

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

Configuring operating system files

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

- For Solaris, add the following line to */etc/pam.conf*:

```
ase auth    required /user/lib/security/$ISA/pam_unix.so.1
```

- For Linux, create a new file called */etc/pam.d/ase*, and add:

```
auth      requried /lib/security/pam_unix.so
```

For more information on how to create these entries, see your operating system documentation.

Running a 32- and 64-bit server on the same machine

$ISA is an environment variable that stands for Instruction Set Architecture. It allows both 32- and 64-bit libraries together.

On Solaris 32-bit machines, $ISA is replaced by an empty string, while on 64bit machines, it is replaced by the string "sparcv9".

If you want to have both 32- and 64-bit, put the 32-bit PAM module in a directory, and put 64-bit version in a subdirectory of this directory.

The entry in *pam.conf* should look similar to:

```
$ ls /usr/lib/security/pam_whatever.so.1
pam_whatever.so.1 ->
/wherever/pam_whatever_32bits.so.1

$ ls /usr/lib/security/sparcv9/pam_whatever.so.1
pam_whatever.so.1 ->
/wherever/pam_whatever_64bits.so.1
```

```
ase    auth    required
/usr/lib/security/$ISA/pam_whatever.so.1
```

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

Configuring Adaptive Server for PAM user authentication

enable pam user auth is a new configuration parameter that enables PAM user authentication support. It can be set as follows:

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

where:

- 0 – disables PAM authentication. This is the default.
- 1 – indicates Adaptive Server will try PAM authentication first, and then fall back to syslogins authentication if PAM authentication fails.
- 2 – indicates only PAM authentication may be used.

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

Enhanced login controls

Adaptive Server version 12.5.2 introduces several new ways to control authentication.

"authenticate with" option

The authentication mechanism is defined when you log in. enable pam user auth and enable ldap user auth enable PAM and LDAP respectively. You can also force a login to use a specific authentication process by using the new options to sp_modifylogin and sp_addlogin.

sp_modifylogin

sp_modifylogin has a new option, authenticate with which has the following values:

- ASE – use Adaptive Server internal authentication using syslogin passwords.
- LDAP – use external authentication with an LDAP Server.
- PAM – use external authentication with PAM.
- ANY – by default, users are authenticated using this authentication method. A user with ANY authentication means that Adaptive Server checks if there is any external authentication mechanism defined, and if there is, it is used. Otherwise, it uses ASE authentication.

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

  - LDAP.
  - PAM.
  - If neither PAM nor LDAP is enabled, ASE authentication through syslogins is used.

  If any of these options are enabled, it is the mechanism used, and no other is tried. For this reason, if both LDAP and PAM are enabled, PAM is never tried for a user with ANY authentication.

Only a System Security Officer with sso_role can set authenticate with for a login.

For example:

```
sp_modifylogin "nightlyjob", "authenticate with", "ASE"
sp_displaylogin "nightlyjob"
```

displays something that looks similar to:

```
Suid: 1234
Loginname: nightlyjob
Fullname: Batch Login
Default Database: master
[...]

Date of Last Password Change: Oct 2 2003 7:38 PM
Password expiration interval: 0
Password expired: N
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ASE
```

sp_addlogin

sp_addlogin accepts a new parameter, @auth_mech which defines the authentication mechanism.

The syntax is:

```
sp_addlogin login, passwd [, defdb]
          [, deflanguage] [, fullname] [, passwdexp]
         [, minpwdlen] [, maxfailedlogins] [, auth_mech]
```

auth_mech can take the same values as sp_modify login "authenticate with" option.

In the following example, individual users can be configured to override global authentication mechanisms:

```
sp_addlogin mylogin, mypassword, @auth_mech = ASE
```

sp_displaylogin

sp_displaylogin now includes output showing the specified authentication mechanism, if any. For example:

```
1> sp_dislaylogin mylogin
2> go
```

now displays something similar to the following:

```
Suid: 1234
Loginname: mylogin
Fullname: My Full Name
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Date of Last Password Change: Oct 2 2003 7:38PM
Password exiration interval: 0
Password expired: N
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ASE
```

Mapping logins using *sp_maplogin*

You can map logins using sp_maplogin using this syntax:

```
sp_maplogin (authentication_mech | null),
```

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

where:

- *authentication_mech* is one of the valid values specified for authenticate with option in sp_modifylogin.
- *client_username* is an external user name. This user name can be an operating system name, a user name for an LDAP server, or anything else that the PAM library can understand. A null value indicates that any login name is valid.
- *action* indicates create login or drop. When create login is used, the login is created as soon as the login is authenticated. drop is used to remove logins.
- *login_name* is an Adaptive Server login that already exists in syslogins.

The following example maps external user “jsmith” to Adaptive Server user “guest”. Once authenticated, “jsmith” gets the privileges of “guest”. The audit login record shows both the *client_username* and the Adaptive Server user name:

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

The following example tells Adaptive Server to create a new login for all external users authenticated with PAM, in case a login does not already exist:

```
sp_maplogin PAM, NULL, "create login"
```

sp_helpmaplogin

sp_helpmaplogin displays mapping information. With no parameters, all login map information displays. You can restrict the output to specific sets of client user names or authentication mechanists by using:

```
sp_helpmaplogin [ (authentication_mech | null),
(client_username | null) ]
```

For example, issuing sp_helpmaplogin returns something that looks similar to:

```
sp_helpmaplogin
authentication   client name   login name
--------------   -----------   ------------------
NULL                  jsmith              guest
PAM                   NULL                create login
```

New global variable @@authmech

Adaptive Server version 12.5.2 includes the new global variable *@@authmech*, which is a read-only global variable, set to the mechanism used to authenticate the user.

For example, consider Adaptive Server is enabled for PAM user authentication with failover (`enable pam user auth = 1`), and Joe is an external user with ANY authentication, and LDAP user authentication is disabled. When Joe logs in, Adaptive Server attempts to authenticate Joe, using PAM user authentication.

If Joe fails authentication as a user in PAM, Adaptive Server authenticates Joe using ASE authentication (syslogins), and if that succeeds, he logs in successfully.

The *@@authmech* global variable now has this value.:

```
1> select @@authmech
2> go

----------------------------------
ase
```

Now consider Adaptive Server is configured for strict PAM user authentication (`enable pam user auth = 2`) and Joe is added as a valid user in PAM. Now, when Joe logs in, the value for *@@authmech* is:

```
1> select @@authmech
2> go

----------------------------------
pam
```

System table changes

The status column from syslogins, a 16-bit integer datatype, has new bits assigned as follows:

Table 7-2: status column from syslogins table

Decimal	Hex	Status
1	0x1	Password contains fewer than six characters or is NULL.
2	0x2	Account is locked.
4	0x4	Password has expired.
8	0x8	User has RepSrv authorization.
16	0x10	OMNI:autoconnect mode is enabled.
32	0x20	Use Adaptive Server internal authentication mechanism – syslogins.
64	0x40	Use only LDAP external authentication.
128	0x80	Use only PAM external authentication.

Access control

The access control changes to Adaptive Server version 12.5.2 are discussed in this section.

Improved granularity for set proxy

In earlier versions of Adaptive Server, set proxy allowed you to switch your server user identity to any other server login, but it did not allow you to limit use of set proxy based on the target login roles. When you granted set proxy to a user, this user could become any other server user.

As of Adaptive Server version 12.5.2, you can grant set proxy...restricted role, which allows you to restrict which roles cannot be acquired when switching identities.

For more information about set proxy, see Chapter 12, “Changes to Global Variables, Commands, and Stored Procedures.”

Grant revoke on administration commands

Adaptive Server version 12.5.2 allows you to grant and revoke permissions for users, roles, and groups for the update statistics, delete statistics, and truncate table commands. Table owners can also provide permissions through an implicit grant by adding update statistics, delete statistics, and truncate table to a stored procedure and then granting execute permissions on that procedure to a user or role.

For more information, see Chapter 12, "Changes to Global Variables, Commands, and Stored Procedures."

Restricted permission on system catalogs

Adaptive Server version 12.5.2 adds the default permissions parameter to the grant and revoke commands, which allows you to grant and revoke the default permissions for some system tables.

For more information, see Chapter 12, "Changes to Global Variables, Commands, and Stored Procedures."

Accountability

audit_event_name is a new function that returns a description of an audit event.

For more information, see Chapter 12, "Changes to Global Variables, Commands, and Stored Procedures."

Encryption

Adaptive Server uses encryption technology to secure communications with clients. Beginning with version 12.5.2, Adaptive Server uses FIPS-certified algorithms for SSL encryption. Adaptive Server version 12.5.2 can also protect backups using passwords.

FIPS certified algorithms for SSL encryption

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

SSL implementation now uses FIPS 140-2 certified algorithms. These algorithms are available through:

- FIPS-140-2 Certified Crypto Module on Windows (Certicom's SB/GSE – NIST Certificate #316 with validation dates of 05/13/2003 and 06/30/2003)
- FIPS certified algorithms on Solaris 32- and 64-bit platforms through Certicom's SB Crypto-C.

Password-protected backups

The Adaptive Server 12.5.2 version of dump and load database include a password parameter that allows you to password-protect your database dumps.

You can protect your database dump from unauthorized loads with the password parameter of the dump database command. If you include the password parameter when you make a database dump, you must also include this password when you load the database.

For more information, see Chapter 9, "Making password-protected database dumps."

CHAPTER 8

Making compressed database dumps

The Adaptive Server version 12.5.2 version of dump includes a compression parameter that allows you to compress your database dumps.

Compressing database dumps

The compression parameter of the dump command allows you to reduce your space requirements for your archived databases. Earlier versions of dump database using the compression API allowed you only to compress your database dumps to a local file. With Adaptive Server 12.5.2, the compression parameter enables you to compress your dumps to a remote machine.

You need not include the compression level when you load the database dump. However, you can issue load with listonly=full to determine the compression level at which the dump was made.

The partial syntax for dump database is:

```
dump database database_name to file_name [ with compression = compress_level]]
```

where:

- *database_name* – is the name of the database from which you are copying data. The database name can be specified as a literal, a local variable, or a stored procedure parameter.
- *file_name* – is the name of the dump file. The name cannot exceed 17 characters, and must conform to operating system conventions for file names.
- *compress_level* – is a number between 1 and 9, with 9 providing the highest level of compression. There is no default compression level; if you do not specify a *compress_level*, Adaptive Server does not compress the dump.

For example, the following dumps the pubs2 database to the remote machine called “remotemachine” and uses a compression level of 4:

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
with compression = "4"
```

CHAPTER 9

Making password-protected database dumps

The Adaptive Server 12.5.2 version of dump and load database include a password parameter that allows you to password-protect your database dumps.

Dumping and loading databases with password protection

You can protect your database dump from unauthorized loads using the password parameter of the dump database command. If you include the password parameter when you make a database dump, you must also include this password when you load the database.

The partial syntax for the password-protected dump database and load database commands are:

```
dump database database_name to file_name [ with passwd = password ]
load database database_name from file_name [ with passwd = password ]
```

where:

- *database_name* – is the name of the database that is being dump or loaded.
- *file_name* – is the name of the dump file.
- *password* – is the password you provide to protect the dump file from unauthorized users.

Your password must be between 6 and 30 characters long. If you provide a password that is less than 6 or greater than 30 characters, Adaptive server issues an error message. If you issue an incorrect password when you attempt to load the database, Adaptive Server issues an eror message and the command fails.

For example, the following uses the password "bluesky" to protect the database dump of the pubs2 database:

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

The database dump must be loaded using the same password:

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

Passwords and earlier versions of Adaptive Server

You can use the password-protected dump and load commands only with Adaptive Server version 12.5.2 and later. If you use the password parameter on a dump of a 12.5.2 version of Adaptive Server, the load fails if you try to load it on an earlier version of Adaptive Server.

Passwords and character sets

You can load the dump only to another server with the same character set. For example, if you attempt to load a dump from a server that uses an ASCII character set to a server that uses a non-ASCII character set, the load fails because the value of the ASCII password is different from the non-ASCII password.

Passwords entered by users are converted to Adaptive Server's local character set. Because ASCII characters generally have the same value representation across character sets, if a user's password is in an ASCII character set, the passwords for dump and load are recognized across all character sets.

CHAPTER 10

Configuring Linux Adaptive Server for Failover on Veritas 2.1

This chapter discusses how to configure Linux Adaptive Server for failover on Veritas Cluster Server (VCS), version 2.1.

Make sure you read the Veritas user manuals and familiarize yourself with the Veritas cluster before you implement the steps in this chapter.

Hardware and operating system requirements

High availability requires the following hardware and system components:

- Two homogenous, networked systems, preferably with similar configurations in terms of resources like CPU, memory, and so on. Adaptive Server supports VCS on RedHat Enterprise Linux 2.1 Advanced Server and VCS version 2.1.

  You should also install the VCS GUI (graphical user interface) to facilitate configuration and administration.

- The two systems must have access to shared multihost disks, which store the databases for the Adaptive Server configured for high availability.

- Install Veritas Volume Manager 3.2 to manage disks and create resources like DiskGroup and Volume.

- Use third-party vendor mirroring for media failure protection.
- Create a service group on each system. A service group is a set of resources that provides a specific service. To provide a service for an Adaptive Server that is configured for high availability, the service group should include such resources as DiskGroup, Volume, Mount, IP, NIC, and so on for Adaptive Server. A sample service group and the resource dependency graph is shown in Figure 10-1 on page 73. See the *Veritas Cluster Server User's Guide* for more information on how to create a service group and how to add resources to a service group.

  Note Each service group must contain at least two resources with one resource of type *HAase* for VCS 2.1. Use the cluster command to establish resource dependency so that the resource type depends on the other resources.

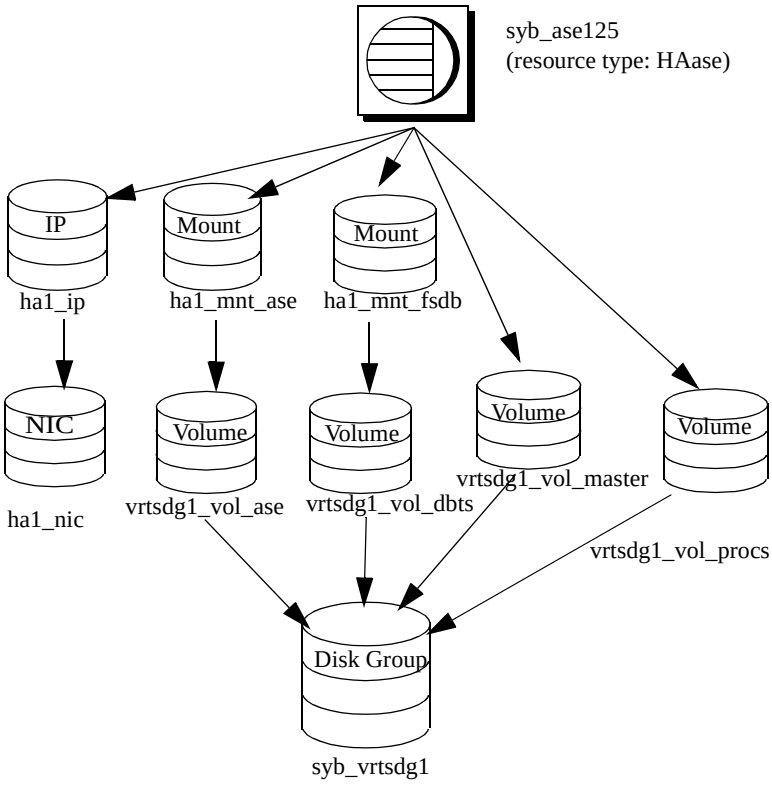
- Configure both public and private networks on both the nodes.

See your hardware and operating system documentation for information about installing platform-specific high availability software.

In Figure 10-1, the configuration of the service group has one DiskGroup, *syb_vrtsdg1*, on which four volumes are created. One volume is for the Adaptive Server installation, one is for databases that are created on the file system, and the other two are for databases created on raw devices. The two mount resources are created for file system of type vxfs (Veritas file system) layering on the volume resources. The resource, *syb_ase125* of type *HAase* is the Adaptive Server installation, which sits on top of the mount resources. *syb_ase125* also requires resource IP, which also requires resource NIC for public network access.

The service group *SybASE* runs on the primary node and another service group, *SybASE2* (not included in Figure 10-1) runs on the secondary node, with a similar configuration.

Figure 10-1: Sample service group running on Veritas Cluster Server



Preparing Adaptive Server to work with the HA subsystem

Perform the tasks in this section to prepare Adaptive Server for a high availability configuration.

Installing Adaptive Servers

Install both the primary and the secondary servers. They can be installed on either shared or local disks. The primary companion can be either a newly installed Adaptive Server, or it can be upgraded from an earlier version of Adaptive Server with existing databases, users, and so on. The secondary companion must be a newly installed Adaptive Server and cannot have any user logins or user databases, to ensure that all user logins and database names are unique within the cluster. After configuration for failover is complete, you can add user logins and databases to the secondary companion.

If you are installing on the local disk, make sure all database devices are created on the multihost disk.

See the installation documentation for your platform for information about installing and configuring Adaptive Server.

Adding entries for both Adaptive Servers to the *interfaces file*

The *interfaces file* for both primary and secondary companion must include entries for both companions. For example, the *interfaces file* for the servers used in the examples used in this chapter would have entries for both MONEY1 and PERSONNEL1. The server entry in the *interfaces file* must use the same network name that is specified in sysservers. For information about adding entries to the *interfaces file*, see the installation documentation for your platform.

Adding entries to the *interfaces file* for client connections during failover

To enable clients to reconnect to the failed over companion, you must add a line to the *interfaces file*. By default, clients connect to the port listed in the query line of the server entry. If that port is not available (because that server has failed over), the client connects to the server listed in the *hafailover* line of the server entry. Here is a sample *interfaces file* for a primary companion named MONEY1 and a secondary companion named PERSONNEL1:

```
MONEY1
master tcp ether MONEY 9678
query tcp ether MONEY 9678
hafailover PERSONNEL1

PERSONNEL1
master tcp ether PERSONNEL 9679
```

```
query tcp ether PERSONNEL 9679
```

Use dsedit to add entries to the *interfaces file*. If the interfaces entries already exist, you must modify them to work for failover.

See the *Adaptive Server Enterprise Utility Guide* for information about dsedit.

The sybha executable

The *sybha* executable allows the Adaptive Server High Availability Basis Services library to interact with each platform's high availability cluster subsystem. The Adaptive Server High Availability Basis Services library calls *sybha*, which is located in *$SYBASE/ASE-12_5/bin*. Before *sybha* can run, you must change its ownership and permissions. You must also edit a file named *sybhauser* in *$SYBASE/ASE-12_5/install*. *sybhauser* contains a list of the users who have System Administrator privileges on the cluster. Sybase strongly recommends that you limit the number of users who have System Administrator privileges on the cluster.

As root, perform the following:

1. Add a new group named *sybhagrp*. You can either add this group to the */etc/group* file, or you can add it to your NIS maps. Add the sybase user to this group (this is the user that owns the *$SYBASE* directory). When the server is started, the sybase user runs the data server. If you have multiple servers running and different users owning the *$SYBASE* directory for each of them, each of these users must be added to the group
2. Change to the *$SYBASE/$SYBASE_ASE/bin* directory:

   ```
   cd $SYBASE/$SYBASE_ASE/bin
   ```

3. Change the ownership of *sybha* to root:

   ```
   chown root sybha
   ```

4. Change the group for the *sybha* program to *sybhagrp*:

   ```
   chgrp sybhagrp sybha
   ```

5. Modify the file permissions for *sybha* to 4550:

   ```
   chmod 4550 sybha
   ```

6. Change to the *$SYBASE/$SYBASE_ASE/install* directory:

   ```
   cd $SYBASE/ASE-12_5/install
   ```

7 Add the sybase user to the *sybhauser* file. These logins must be in the format of UNIX login IDs, not Adaptive Server logins. For example:

```
sybase
coffeecup
spooner
venting
howe
```

8 Change the ownership of *sybhauser* to root:

```
chown root sybhauser
```

9 Modify the file permissions for *sybhauser*:

```
chmod 600 sybhauser
```

Creating a new default device

By default, master is the default device in a newly installed Adaptive Server. This means that, if you create any databases (including the proxy databases used by failover), they are automatically created on the master device. Adding user databases to the master device makes it difficult to restore the master device from a system failure. To make sure that the master device contains as few extraneous user databases as possible, create a new device using disk init. Use sp_diskdefault to specify the new device as the default before you configure Adaptive Server as a companion for failover.

For example, to add a new default device named money_default1 to the MONEY1 Adaptive Server, enter:

```
sp_diskdefault money1_default1, defaulton
```

The master device continues to also be a default device until you specifically issue the following to suspend it as the default device:

```
sp_diskdefault master, defaultoff
```

See the *Adaptive Server Reference Manual* for more information about disk init and sp_diskdefault.

Adding the local server to *sysservers*

Use sp_addserver to add the local server in sysservers using the network name specified in the *interfaces file*. For example, if the companion MONEY1 uses the network name of MONEY1 in the *interfaces file*:

```
sp_addserver MONEY1, local, MONEY1
```

You must reboot Adaptive Server for this change to take effect.

Adding secondary companion to *sysservers*

Add the secondary companion as a remote server in *sysservers*:

```
sp_addserver server_name
```

By default, Adaptive Server adds the server with a srvid of 1000. You do not need to reboot Adaptive Server for the change to take effect.

Assigning *ha_role*

You must have the ha_role on both Adaptive Servers to run sp_companion. To assign the ha_role, issue the following from isql:

```
sp_role "grant", ha_role, sa
```

You must log out and then log back in for the changes to Adaptive Server to take effect.

Installing HA stored procedures

Note You must already have added both servers to the *interfaces* file before you can install the high-availability stored procedures. If you run *installhasvss* before performing these tasks, you must reinstall all the system stored procedures.

The *installhasvss* script:

- Installs the stored procedures required for failover (for example, sp_companion).
- Installs the *SYB_HACMP* server in sysservers.

You must have System Administrator privileges to run *installhasvss*.

installhasvss is located in *$SYBASE/ASE-12_5/scripts*. To execute *installhasvss*, enter:

```
$SYBASE/$SYBASE_OCS/bin/isql -Usa -Ppassword -
Sservername < $SYBASE/ASE-12_5/scripts/installhasvss
```

installhasvss prints messages as it creates stored procedures and creates the SYB_HACMP server.

Verifying configuration parameters

You must enable the following configuration parameters before you configure Adaptive Server for Failover:

- enable CIS – enables Component Integration Services (CIS). This configuration parameter is enabled by default.
- enable xact coordination – enables Distributed Transaction Management (DTM). This configuration parameter is enabled by default.
- enable HA – enables Adaptive Server to function as a companion in a high availability system. enable HA is off by default. You must reboot Adaptive Server for it to take effect. This parameter causes a message to be written to your error log stating that you have started the Adaptive Server in a high availability system. You must purchase the ASE_HA license option to use Adaptive Server with Failover. See the installation guide for your platform for information about enabling the ASE_HA license.

See the *System Administration Guide* for information about enabling configuration parameters.

Adding thresholds to the master log

Failing over, failing back, creating proxy databases and so on are log-intensive activities. If you do not have adequate log space, any of these activities can fail. If you have not already done so, you must add a threshold to the master log.

1. Define and execute sp_thresholdaction on the master database's log to set a threshold on the number of pages left before a dump transaction occurs. Sybase does not supply sp_thresholdaction. See the *Adaptive Server Reference Manual* for information about creating this system procedure.
2. Place thresholds on the master log segment so it does not fill up:

```
sp_addthreshold "master", "logsegment", 250, sp_thresholdaction
```

3. You must reboot the primary companion for this static parameter to take effect.

Configuring the Veritas subsystem for Sybase Failover

This section assumes that you have already installed the high availability subsystem. See the *VCS Installation Guide* and *VCS User's Guide* for information about installing and using the Veritas Cluster Server high availability subsystem.

Installing the agent on VCS version 2.1

Perform the following steps to install the agent on each node of the cluster (you must have root permission to run the following commands):

1 Change to the *$SYBASE/$SYBASE_ASE/install/veritas/HAase* directory:

```
cd $SYBASE/$SYBASE_ASE/install/veritas/HAase
```

2 Execute the installation script:

```
perl installHAase.pl
```

The installation script:

- Copies the *HAase* resource type file *HaaseTypes.cf* to */etc/VRTSvcs/conf/config/* on local system
- Makes a new directory, */opt/VRTSvcs/bin/HAase*, if it does not already exist
- Copies the following agent binary and scripts to */opt/VRTSvcs/bin/HAase/* on the local system:
  - *HAaseAgent*
  - *online*
  - *offline*
  - *clean*
  - *sybhautil.pm*
  - *attr_changed*

Configuring the resource type for VCS version 2.1

Perform the following to install the resource type on VCS version 2.1.

Creating an Adaptive Server login file

Create a file that contains the Adaptive Server login information for System Administrator and for the user you added for the fault monitor. A sample file containing a template for this information is located in: *$SYBASE/$SYBASE_ASE/install/veritas/HAase/ase_login_file*.

This file consists of two lines. The first line is the login and password for the System Administrator; the second line is the monitor user login and password.

```
login-type<tab>login string
login-type<tab>login string
```

The *login-type* and the *login string* must be separated by a `tab` character.

Note If you use another file at a different location, specify the full path for the resource extension property *Dataserver_login_file* when configuring the *HAase* resource.

The default value for *login-type* is `normal`. Values for *login string* are in the form *login-name/password.* For example:

```
normal      sa/sa-password
normal      probe-user/probe-password
```

For security reasons, the *ase_login_file* should be securely protected so that `read` and `write` access permissions are restricted to root. To maintain this security, perform the following:

```
chmod 400 ase_login_file
chown root ase_login_file
chgrp  sys  ase_login_file
```

Note Sybase strongly recommends that you use a non-empty password. If you use an empty password, the agent scripts generate a warning message.

Importing the resource type

There are two ways you can import the *HAase* resource type:

- Use the cluster GUI tool to import the new resource type. See your *VCS User Guide* for more information.

- Use cluster commands hatype and haattr to manually import the new resource type from the command line. See your *VCS User Guide* for more information.

Starting the agent

You can start the agent by either:

- Restarting the Veritas Cluster, or
- Using the cluster commands to manually start the agent

The second method is more desirable since it causes no disruption. The steps to manually start the agent are:

1 Check the status of *HAase* agent with the *haagent* utility:

```
#haagent -display HAase
#Agent   Attribute  Value
HAase     AgentFile
HAase     Faults     0
HAase     Running    No
HAase      Started      No
```

2 Start the *HAase* agent on the host *myhost* with the *haagent* utility:

```
# haagent -start HAase -sys myhost
VCS:10001:Please look for messages in the log file
```

3 Check the status of *HAase* agent using the *haagent* utility:

```
# haagent -display HAase
#Agent    Attribute   Value
HAase     AgentFile
HAase     Faults      0
HAase     Running     Yes
HAase      Started       Yes
```

Adding the resource

Each service group should contain a resource. The following table shows the attributes of an resource.

Property	Datatype, dimension, and default	Description
Sybase_home	string, scalar, null	The home directory of the Adaptive Server installation, and the same as the value for the environment variable SYBASE in an Adaptive Server installation.
Dataserver_name	string, scalar, null	Name of the Adaptive Server that is supplied at the time of configuration.
Backup_server_name	string, scalar, null	Name of the Backup Server that is supplied at the time of configuration.
Monserver_name	string, scalar, null	Name of the Monitor Server that is supplied at the time of configuration.
Textserver_name	string, scalar, null	Name of the full-text search server that is supplied at the time of configuration.
Secondary_companion_name	string, scalar, null	Name of secondary companion server which is set when you run the 'sp_companion configure' command.
Dataserver_login_file	string, scalar, null	Absolute path to a file containing login information for the current data server. The file consists of two lines; the first line is the login and password for the System Administrator, the second line is the user login and password used for thorough probe used by the HA agent monitor.
RUN_server_file	string, scalar, null	Absolute path to an alternative *RUN_server* file, which overwrites the default *$SYBASE/$SYBASE_ASE/install/RUN_SERVER*.
Thorough_probe_cycle	int, scalar, 3	The number of "shallow" probes before a thorough probe is performed.

Property	Datatype, dimension, and default	Description
Thorough_probe_script	string, scalar, null	Absolute path to an alternative file containing SQL scripts for the fault monitoring program to perform a thorough probe. If it is set to null, the agent uses the default SQL commands. For security reasons, this file should only restrict write access to the owner of *$SYBASE* directory). **Note** This value is ignored by the *HAase* resource.
Debug	Boolean, scalar, 0	If set to 1 (true), the monitor logs debugging messages to *$VCS_LOG/log/HAase_A.log*; other scripts log debugging messages to *$VCS_LOG/log/engine_A.log*. The message number range is 2,000,001 and greater.
Log_max_size	int, scalar, 5000000	Maximum size for the *$VCS_LOG/log/HAase_A.log* file.
Failback_strategy	string, scalar, null	Reserved for future use.
HA_config	boolean, scalar, 0	Reserved for future use.
Cmpstate	boolean, scalar, 0	Reserved for future use.

Note The default value for *$VCS_LOG* is */var/VRTSvcs*.

The following table shows a sample configuration of an *HAase* instance:

Attribute	Value
Sybase_home	*/release/rel125*
Dataserver_name	Money1
Backup_server_name	
Monserver_name	
Textserver_name	
Secondary_companion_name	
Dataserver_login_file	*/release/rel125/ASE-12_5/install/MONEY1_login*

Attribute	Value
RUN_server_file	*/release/rel125/ASE-12_5/install/RUN_MONEY1*
Thorough_probe_cycle	3
Thorough_probe_script	
Debug	
Log_max_size	5000000
Failback_strategy	
HA_config	0
Cmpstate	0

Configuring an instance of *HAase* for each service group

You can configure an instance of the *HAase* by either:

- Using the cluster GUI tool to configure an instance of the resource type. See your *VCS User Guide* for more information), or
- Using cluster commands to manually add a new resource and configure its attributes. This is described below.

  The following are the cluster commands used to add resource *syb_ase125* that uses the configuration described in the table above. (The configuration of service group *SybASE* is shown in Figure 10-1 on page 73):

  a Add the resource type (this example uses the *HAase* resource type):

```
#hares -add syb_ase125 HAase SybASE
VCS:10245:Resource added
NameRule and Enabled attributes must be set before agent monitors
# hares -modify syb_ase125 Dataserver_name MONEY1
# hares -modify syb_ase125 RUN_server_file  /release/rel125/ASE-
12_5/install/RUN_MONEY1
# hares -modify syb_ase125 Log_max_size 5000000
# hares -modify syb_ase125 Dataserver_login_file /release/rel125/ASE-
12_5/install/MONEY1_login
# hares -modify syb_ase125 Sybase_home  /release/rel125
# hares -modify syb_ase125 Thorough_probe_cycle 3
```

  b Configure the agent to monitor the status of resource *syb_ase125*:

```
# hares -modify syb_ase125 Enabled 1
```

Note After you add the new resource to service group, you must establish the resource dependency between the resource type and other storage and network access resources access.

Use the following cluster commands to establish a resource dependency between *syb_ase125* and resources of types *Mount*, *Volume*, and *IP*:

```
# hares -link syb_ase125 ha1_mnt_ase
# hares -link syb_ase125 ha1_mnt_fsdb
# hares -link syb_ase125 vrtsdg1_vol_master
# hares -link syb_ase125 vrtsdg1_vol_procs
#hares -link syb_ase125 ha1_ip
```

Refer to Figure 10-1 for more details.

Configuring companion servers for Failover

Perform the tasks in this section to configure the Adaptive Servers as primary and secondary companions in a high availability system.

Adding user and login for HA monitor on VCS versions 2.1

Create a special user and login for the monitor for each data server associated with the *HAase* resource. Use isql to connect to the data servers and issue:

```
sp_addlogin probe_ase, password

sp_adduser probe_ase
```

For example:

```
sp_addlogin joe, joe_password
```

```
sp_adduser joe
```

Note During Adaptive Server configuration, the System Administrator should take into account that the user and login used for probe actually reduces by one the total number of connections available for other purposes.

Because the applications connecting to Adaptive Server may use all available user connections, the probe_user may not be able to log in to Adaptive Server, which could cause Adaptive Server to fail over. You should set the number of user connections high enough to prevent this from happening.

For more information about storing the monitor login information, see "Creating an Adaptive Server login file" on page 80.

Running *sp_companion* with *do_advisory*

You must configure the secondary companion with sufficient resources to perform the work of both servers during failover. The secondary companion may have attributes that prevent a successful cluster operation. For example, if both the primary and secondary companions are configured for 250 user logins, during failover, the secondary companion only has the resources for half the number of potential user logins necessary. Instead, both MONEY1 and PERSONNEL1 should be configured for 500 user logins.

The sp_companion do_advisory option checks the configuration options on both the primary and the secondary companion to make sure a cluster operation is successful. sp_companion do_advisory advises you of any configuration options that should be changed.

See Chapter 5, "Running do_advisory" for a complete description of the sp_companion do_advisory option.

Verifying the HA agent

Because Adaptive Server can support different cluster software, sp_companion includes the show_cluster option to query the HA agent currently running and the set_cluster option to set the HA agent.

If you are running the Veritas Cluster Server subsystem, you must specify this with sp_companion. Adaptive Server assumes it is running the Sun Cluster software on the Solaris platform, unless you specify otherwise. Set the agent to *VCS-HAase* for VCS versions 2.1.

The syntax is:

sp_companion *companion_server_name*, [show_cluster]
sp_companion *companion_server_name*, [set_cluster ["VCS-HAase"]]

In the following example, Adaptive Server is using the default HA agent for Sun Cluster 2.2:

```
sp_companion MONEY1, show_cluster
The default cluster is: SC-2.2
The current cluster is set to default.
Supported cluster systems for SunOS:
SC-2.2
VCS-Sybase
SC-3.0
VCS-HAase
```

To change the Adaptive Server to use the *HAase* agent for the Veritas Cluster:

```
sp_companion MONEY1, set_cluster, "VCS-HAase"

The current cluster is set to VCS-HAase
```

The Adaptive Server now uses the *HAase* agent for the VCS subsystem.

Note Do not change to another HA agent type when Adaptive Server is configured for normal companion mode on your VCS system.

Configuring for asymmetric configuration

Two Adaptive Servers are configured for asymmetric configuration. From the secondary Adaptive Server, issue:

```
sp_companion "primary_server_name", configure, with_proxydb, login_name,
password
```

Where:

- *primary_server_name* is the name of the primary Adaptive Server as defined in the *interfaces file* entry and in sysservers.

- *with_proxydb* indicates that proxy databases are created on the secondary companion for all databases other than system databases. Any subsequent databases that are added also create proxy databases.
- *login_name* is the name of the user performing this cluster operation (they must have the ha_role).
- *password* is the password of the person performing this cluster operation.

This example configures an Adaptive Server named MONEY1 as a primary companion (issue the command from the secondary companion, PERSONNEL1):

```
sp_companion "MONEY1", configure, null, "Think2Odd", "password"
Server 'PERSONNEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONNEL1' to Server:'MONEY1'.
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONNEL1'.
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
...
(1 row affected)
(1 row affected)
(1 row affected)
Step: Companion server's configuration check succeeded.
Step: Server handshake succeeded.
Step: Master device accessible from companion.
Step: Added the servers 'PERSONNEL1' and 'MONEY1' for cluster configuration.
Step: Server configuration initialization succeeded.
Step: Synchronizing Application Specific information from companion server
Step: Synchronizing Roles from companion server
Step: Synchronizing Login Roles from companion server
Step: Synchronizing Remote Logins from companion server
Step: Synchronizing Groups in sysusers from companion server
Step: Synchronizing Sysattributes from companion server
Step: Synchronizing server logins from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information synchronization succeeded.
Step: Server configured in normal companion mode.
```

If user databases already exist while you are using sp_companion, you see messages similar to these:

```
Step: Created proxy database 'pubs2'
 Step: Proxy status for database has been set. Please Checkpoint the database
'pubs2'
```

```
Step: Server configured in normal companion mode"
Starting companion watch thread
```

See Chapter 3, "Asymmetric and Symmetric Setup" in the *High Availability User's Guide* for more information about asymmetric configuration.

Note The *login_name* and *password* in the above sp_companion configure command cannot be null. After you successfully execute sp_companion configure, the operating system creates a new file, */etc/VRTSvcs/conf/config/ha_companion.remote_server_name*. Make sure this file has read and write access only for the user who runs the server, otherwise, security may be compromised.

Configuring for symmetric configuration

After you configure your companions for asymmetric failover, you can configure them for symmetric configuration. In a symmetric configuration, both servers act as primary and secondary companions.

Issue sp_companion from the primary companion to set it up for symmetric configuration. Use the same syntax as for asymmetric configuration. See "Configuring for asymmetric configuration" on page 87, above, for a description of the syntax for sp_companion.

The following example adds an Adaptive Server named MONEY1 as the secondary companion to the Adaptive Server named PERSONNEL1 (issue this command from primary companion MONEY1):

```
sp_companion "PERSONNEL1", configure, with_proxydb, null, sa, Think2Odd
Server 'MONEY1' is alive and cluster configured.
Step: Access verified from Server:'MONEY1' to Server:'PERSONNEL1'
Server 'PERSONNEL1' is alive and cluster configured.
Step: Access verified from Server:'PERSONNEL1' to Server:'MONEY1'
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
.....
Step: Companion servers configuration check succeeded
Step: Server handshake succeeded
Step: Master device accessible from companion
```

```
Step: Added the servers "MONEY1" and "PERSONNEL1" for cluster config
Step: Server configuration initialization succeeded
Step: Synchronizing server logins from companion server
Step: Synchronizing remoteserver from companion server
Step: Synchronizing roles from companion server
Step: Synchronizing server-wide privs from companion server
Step: User information syncup succeeded
Step: Server configured in normal companion mode
```

Note The *login_name* and *password* in the above sp_companion configure command cannot be null. After you successfully execute sp_companion configure, the operating system creates a new file, */etc/VRTSvcs/conf/config/ha_companion.remote_server_name*. Make sure this file has read and write access only for the user who runs the server, otherwise, security may be compromised.

Administering Sybase Failover

This section includes information about using Sybase Failover.

During failover

When the primary node fails over to the secondary node, the service group that is online on the primary node is switched to the secondary node. At this point, all the resources except the Adaptive Server binary are online on the secondary node. The Adaptive Server on the secondary node takes over these resources.

Note When one service group fails over from the primary host to the secondary host, the Adaptive Server on the secondary host takes over all its resources, but the Adaptive Server on the failed-over group is not started.

Failing back to the primary companion

Failback switches the service group that originally belonged to the primary node from the secondary node back to the primary node and brings it online.

To initiate failback:

1 After your primary node is ready to take back the service group, issue the following from the secondary companion:

```
sp_companion primary_companion_name,
prepare_failback
```

where *primary_companion_name* is the name of primary companion. This command switches the primary node's service group from secondary node back to primary node. For example, to fail back the primary companion MONEY1, issue this command from the secondary companion PERSONNEL1:

```
sp_companion "MONEY1", prepare_failback
Step: Primary databases are shutdown in secondary.
Step: Primary databases dropped from current secondary.
Step: Primary devices released from current secondary.
Step: Prepare failback for primary server completed successfully.
```

2 Make sure the primary nodes service group is successfully switched to primary node by issuing this command from the command line:

```
hastatus -group service_group_name
```

This command displays the status of the primary nodes service group.

3 To resume normal companion mode, issue the following from the primary companion:

```
sp_companion secondary_companion_name, resume
```

where *secondary_companion_name* is the name of the secondary companion server. For example, to resume normal companion mode for primary companion MONEY1:

```
sp_companion "PERSONNEL1", resume
```

Note You cannot connect clients with the failover property (for example isql -Q) to Adaptive Server until you issue sp_companion resume. If you do try to reconnect them after issuing sp_companion prepare_failback, the client hangs until you issue sp_companion resume.

Suspending normal companion mode

Suspended mode temporarily disables the ability of the primary companion to fail over to the secondary companion. To switch from normal companion mode to suspended mode:

1 As root, use hares to change the attribute *Critical* for the *Sybase* resource on primary node to 0. The syntax is:

```
hares -modify name_of_Sybase_resource Critical 0
```

For example, to modify the attribute *Critical* of the *Sybase* resource, *syb_ase125* for primary companion, MONEY1:

```
hares -modify syb_ase125 Critical 0
```

(See the *Veritas Cluster Server User's Guide* for more information about the hares command.)

2 Suspend normal companion mode. From the secondary companion, issue:

```
sp_companion companion_name, suspend
```

For example, to suspend primary companion MONEY1 for maintenance, connect to secondary companion PERSONNEL1 and issue:

```
sp_companion MONEY1, suspend
```

Resuming normal companion mode

To move from suspended mode to normal companion mode:

1 Make sure both companions are running. As root, issue:

```
hastatus
```

2 Change the *Critical* attribute of the *Sybase* resource on the primary node to 1. As root, issue:

```
hares -modify name_of_Sybase_resource Critical 1
```

For example, to modify the *Critical* attribute of the *Sybase syb_ase125* resource for primary companion MONEY1:

```
hares -modify syb_ase125 Critical 1
```

3 Resume normal companion mode. From the secondary companion, issue:

```
sp_companion primary_companion_name, resume
```

For example, to resume normal companion mode for primary companion MONEY1:

```
sp_companion MONEY1, resume
```

Note You cannot connect clients with the failover property (for example isql -Q) until you issue sp_companion resume. If you do try to reconnect them after issuing sp_companion prepare_failback, the client hangs until you issue sp_companion resume.

Dropping companion mode

To drop companion mode, issue:

```
sp_companion companion_name, "drop"
```

Dropping companion mode is irreversible; you must reconfigure the companion servers before they fail over in a high availability system and retain all the functionality that Sybase Failover provides. However, the companion server is still monitored by the HA agent. Before dropping companion mode, you must first disable the agent to monitor Adaptive Server. Issue the following command:

```
hares -modify Sybase_resource_name Enabled 0
```

To drop the companion mode, issue sp_companion ... drop.

For example, to drop the companionship with primary companion MONEY1, connect to secondary companion, PERSONNEL1 and issue:

```
sp_companion "MONEY1", "drop"
```

Troubleshooting Failover for Veritas Cluster

This section includes troubleshooting information about common errors.

- Turn on the debugging for Adaptive Server. Use the trace flag 2205 to get high availability-related debugging information. The following isql session turns on the debugging and redirects its messages to the console:

  ```
  dbcc traceon(2205)
  dbcc traceon(3604)
  ```

- When your system reports errors, first check the error log. In the VCS system log, */var/VRTSvcs/log/engine_A.log*, any error message with a message ID greater than 2,000,000 is an error message from *HAase* agent.
- The VCS error logs are located in, */var/VRTSvcs/log/log_name.log*. Among them, the *engine_A.log* is an important source of information.

  The system error log is located in */var/log/syslog*.
- Sybase recommends that you use the following monitoring tools to find information about your system:
  - *hagui* – a GUI tool.
  - *hastatus* – a command line tool.
  - The following trigger scripts, which alert you of events on the VCS system: *injeopardy*, *preonline*, *postonline*, *postoffline*, *resnotoff*, *resfault*, *sysoffline*, *violation*.
- When one service group fails over from the primary host to the secondary host, the Adaptive Server on the secondary host takes over all its resources, but the Adaptive Server on the failed-over group is not started, and VCS may indicate that the HAase resource is "faulted" on the secondary host. Use the following command on the secondary host to clear the state after failover:

  ```
  hares -clear sybase_res_name -sys
  secondary_host_name
  ```

Recovering from a failed *prepare_failback*

During a failback, if prepare_failback was executed successfully on the secondary companion but the primary companion does not boot:

1. Check the primary companion's error log and the cluster error log to identify why the server did not boot, and correct the problems.
2. To clear the FAULTED state of the resource type, issue:

   ```
   hares -clear HAase_res_name
   ```
3. As root, issue the following to move the primary logical host back to the secondary node:

   ```
   hagrp -switch primary_service_group -to
   secondary_host_name
   ```
4. Log in to the secondary companion and issue:

```
dbcc ha_admin ("", "rollback_failback")
```

Your companion servers should both be back in failover mode. For more information about dbcc ha_admin, see “dbcc Options for High Availability Systems on page 354 in the *High Availability User’s Guide*.

5 Reissue sp_companion...prepare_failback on the secondary companion.

Location of the logs

Use the information in these logs to debug your high availability subsystem:

- Adaptive Server error log (defined in the RUNSERVER file).
- Veritas cluster log, located in */var/VRTSvcs/log/engine_A.log*.
- Operating system messages are in */var/log/syslog*.
- *HAase* agent log on VCS versions 2.1, located in */var/VRTSvcs/log/HAase_A.log.*

CHAPTER 11

Large Memory Support for 32-Bit Linux

Overview

Adaptive Server large memory support on 32-bit Enterprise Linux operating systems increases the amount of available memory in Adaptive Server from 2.7GB to as much as 64GB. Increasing the amount of memory available to Adaptive Server improves performance by significantly reducing the number of times the server must access the disk.

Adaptive Server large memory support makes use of the shared memory file system (*shmfs*) and the memory mapped file (*mmap*) features available on Linux 32-bit operating systems. When large memory support is enabled, Adaptive Server creates an *shmfs* file of a size specified in the Adaptive Server configuration file. This file can be up to 16GB for Red Hat Advanced Server Linux 2.1 (AS 2.1) operating systems and up to 64GB for Red Hat Enterprise Linux 3.0 (RHEL 3) operating systems.

Adaptive Server uses memory-mapped file space for a secondary data cache. The primary data cache can only be as large as the 2.7GB, which is the maximum addressable memory allowed by Linux, minus any necessary overhead.

Adaptive Server always places new pages in the primary data cache. When the primary cache is full, Adaptive Server relocates a page to the secondary cache to make room for the new page. Adaptive Server searches for a given page first in the primary data cache and then in the secondary cache. If the page is found in the secondary cache, Adaptive Server maps a virtual address window to that portion of the *shmfs*, and copies the page to the primary cache where it can be read.

The secondary cache:

- Is global; it is shared by all named primary caches.
- Can replace only pages of the same size as the server page size. Thus, if the server page size is 4K and there are two pools, one with a page size of 4K and the other with a page size of 16K, the secondary cache holds only pages from the 4K pool.
- Supports writes only for washing and checkpointing.
- Has a fixed wash size of 20%. Wash size for pools in primary caches are not honored—all washes take place in the secondary cache.
- Does not support index trips and OAM trip tuning.
- Will not hold pages from caches marked “log only” or “relaxed lru” because such caches do not benefit from the secondary cache. Log pages are typically not read back from disk, and caches are marked “relaxed lru” only when the object is fully cached in the primary cache.

Setting up large memory support

To set up large memory support:

1 Configure the operating system for *shmfs*.

2 Configure Adaptive Server for the secondary data cache.

Configuring the operating system

Use operating system commands to configure the server for *shmfs* and specify its size.

For example, to configure an *shmfs* of 8GB on either Linux AS 2.1 or Linux RHEL 3, enter:

```
mount -t shm shmfs -o size=8g /dev/shm
```

Note Make sure that the Sybase user has read and write permissions on */dev/shm*.

See your Linux operating system documentation for more information.

Configuring Adaptive Server

To configure Adaptive Server for the secondary cache, enter a value for the extended cache size configuration parameter. When the value of extended cache size is zero (the default), Adaptive Server does not use a secondary cache.

To enable the secondary cache, specify a size for the secondary cache in 2K pages. For example, to create a secondary cache of 1048576 2K pages (about 2GB) using sp_configure, enter:

```
sp_configure "extended cache size", 1048576
```

When the secondary cache is enabled, Adaptive Server creates the file */dev/shm/<server_name><pid_number>.shm*.

If a secondary data cache is enabled:

- Adaptive Server deletes the secondary cache whenever Adaptive Server shuts down, or the value of extended cache size is reset to zero.
- Adaptive Server creates a secondary cache whenever Adaptive Server starts up.
- Zero is the default value. When the value of extended data cache is zero, Adaptive Server does not use the secondary data cache.

Changing the size of the secondary data cache

You can change the size of the secondary data cache.

Increasing the size of the secondary cache

To increase the size of the secondary data cache to the declared size of the *shmfs*, use sp_configure. This is a dynamic change and does not require restarting Adaptive Server. For example, to increase cache size to 8GB, enter:

```
sp_configure "extended cache size", 4194304
```

To increase the size of the secondary cache *beyond* the *shmfs* size configured by the operating system:

1 Disable the secondary cache. Enter:

```
sp_configure "extended data cache", 0
```

2 Unmount the */dev/shm*.

3 Mount the *shmfs* with the desired size.

See "Configuring the operating system" on page 98 for instructions.

4 Enable the secondary data cache with the desired size.

See "Configuring Adaptive Server" on page 99 for instructions.

Decreasing the size of the secondary cache

You can decrease the size of the secondary cache in two ways: statically, which requires a reboot of Adaptive Server, and dynamically, which does not.

To statically decrease the size of the secondary data cache:

1 Use sp_configure to reset the size of the secondary data cache to the desired size. For example, to increase cache size to 2GB, enter:

```
sp_configure "extended cache size", 1048576
```

2 Reboot Adaptive Server.

To dynamically decrease the size of the secondary data cache:

1 Disable the current secondary cache. Enter:

```
sp_configure "extended data cache", 0
```

2 Re-create the secondary cache in the desired size. For example:

```
sp_configure "extended cache size", 1048576
```

System stored procedure changes

Although the user interface has not changed, these system stored procedures have been modified for large memory support:

- sp_configure
- sp_helpconfig
- sp_sysmon

Changes to sp_configure

The extended cache size parameter has been added to the list of Adaptive Server parameters configurable by sp_configure. Use extended cache size to specify the size of the secondary cache. Setting extended cache size to zero (0), which is the default value, disables the secondary cache.

See “Configuring Adaptive Server” on page 99 for an example.

Changes to sp_helpconfig

sp_helpconfig displays information about configuration parameters. Use sp_helpconfig to calculate the overhead associated with enabling the secondary data cache.

For example, to calculate the overhead for a secondary cache of 2GB (1048576 2K pages), enter:

```
sp_helpconfig “extended cache size”, “1048576”
```

Changes to sp_sysmon

sp_sysmon displays performance information about Adaptive Server. Use sp_sysmon to display performance information about the secondary data cache. This information is displayed in the Data Cache Management section.

For example:

```
Cache Statistics Summary (All Caches)
-------------------------------------
                                  per sec    per xact    count    # of total
                                  ----------------------------------------
...


Secondary Cache Search Summary
       Total Cache Hits          152448.5     432.2    15286912      100.0 %
```

```
       Total Cache Misses            26.6      28.0        1597        0.0 %
--------------------------      --------    -------  ---------
    Total Cache Searches         152475.2     450.3    15288509
```

extended cache size configuration parameter

The extended cache size configuration parameter appears in the Cache Manager section of the configuration file. For example:

```
[Cache Manager]
   number of oam trips = DEFAULT
   number of index trips = DEFAULT
   memory alignment boundary = DEFAULT
   global async prefetch limit = DEFAULT
   global cache partition number = 4
   extended cache size = 4194304
```

Cache Manager

extended cache size

Summary information	
Default value	0
Range of values	0 – size of shared memory file system (*shmfs*) in 2K pages
Status	Dynamic – when increasing cache size Static – when decreasing cache size
Display level	Comprehensive
Required role	System Administrator

The extended cache size parameter specifies the size of the global secondary data cache for 32-bit Enterprise Linux operating systems. Specifying a secondary data cache increases the amount of memory available to Adaptive Server, and improves performance by reducing the need to access the disk. Without the secondary data cache, Adaptive Server is limited to the 2.7GB of available memory allowed by Linux.

To create a secondary data cache, operating system commands must first be used to create a shared memory file system (*shmfs*) of up to 16GB for AS 2.1 operating systems and 64GB for RHEL 3 operating systems. Adaptive Server creates the secondary cache file when:

- The secondary cache is enabled by setting extended cache size to a value greater than zero.
- Adaptive Server starts up and the value of extended cache size is greater than zero.

When the value of extended cache size is zero, Adaptive Server does not use or create a secondary data cache.

CHAPTER 12

Changes to Global Variables, Commands, and Stored Procedures

This chapter describes new and changed global variables, functions, commands, and stored procedures to Adaptive Server version 12.5.2.

New global variables

Table 12-1: New global variables

Global variable	Definition
@@monitors_active	Reduces the number of messages displayed by sp_sysmon.
@@authmech	A read-only variable that indicates the mechanism used to authenticate the user.

The Real Time Meassages Sevices option adds a number of global variables as well. See the *Real Time Data Service User's Guide* for more information.

New configuration parameters

This section discusses new configuration parameters for Adaptive Server version 12.5.2.

The Real Time Meassages Sevices option adds a configuration parameter as well. See the *Real Time Data Service User's Guide* for more information.

histogram tuning factor

Summary information	
Default value	1 (off)
Range of values	1 – 100
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

histogram tuning factor controls the number of steps Adaptive Server analyzes per histogram for update statistics, update index statistics, update all statistics, and create index.

In the following example, Adaptive Server generates an intermediate 20-step histogram with 30 values:

```
sp_configure 'histogram tuning factor',20
update statistics tab using 30 values
```

Adaptive Server analyzes the histogram and compresses it into the resulting histogram according to the following parameters:

- The first step is copied unchanged.
- The high-frequency steps are copied unchanged.
- The consecutive range steps are collapsed into the resulting step, so the total weight of the collapsed step would not be bigger than one-thirtieth of the value.

The final histogram in sysstatistics:

- Has range steps generated in a way similar for a 30-step update statistics, and high frequency ranges are isolated as if the histogram were created with 600 steps.
- The total number of steps in the resulting histogram may differ between 30 and 600 values.
- For equally distributed data, the value should be very close to 30.
- More "frequent" values in the table means more steps in the histogram.

- If a column has few different values, all those values may appear as high-frequency cells.

You could achieve the same result by increasing the number of steps to 600 as using histogram tuning factor, but this would use more resources in the buffer and procedure cache

historgram tuning factor minimizes the resources histograms consume, and only increases resource usage when it is in the best interest for optimization. For example, when there is non-uniform distribution of data in a column, or highly duplicated values within a column. In this situation, up to 600 histogram steps are used. However, in most cases, it uses the default value (30 in the example above).

number of dump threads

Summary information	
Default value	Disabled
Range of values	1 (disabled, no parallelism) – 8 (fully parallel)
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

number of dump threads controls the number of threads that Adaptive Server spawns to perform a memory dump. Using the appropriate value for number of dump threads can reduce the amount of time the engines are halted during the memory dump.

Consider the following when you are determining the number of threads for memory dumps:

- Use a value of 8 if the machine has enough free memory for the file system cache to hold the entire memory dump.
- If you do not know whether the machine has enough free memory, the value for number of dump threads depends on many factors, including the speed of the I/O system, the speed of the disks, the controller's cache, whether the dump file lives in a logical volume manager created on several disks, and so on.
- Use a value of 1 (no parallel processing) if you do not halt the engines when performing memory dumps, described below.

When Adaptive Server performs a memory dump, the number of files it creates is the sum of the number of memory segments that it has allocated multiplied by the number of threads configured. Adaptive Server uses separate threads to write on separate files. When this job completes, the engines are restarted, and the files are merged into the target dump file. Because of this, the time to dump the shared memory in parallel is greater than doing it serially.

- If you halt the engines during the memory dump, a value other than 1 may reduce the amount of time the engines spend stopped while dumping the memory.

New sort orders

Adaptive Server version 12.5.2 adds the following sort order.

Table 12-2: New sort orders

Language or script	Character set	Sort orders
Czech and Slovic	cp852, iso88592, and cp1250	dictionary, nocase, noaccents
Western European	cp1252	dictionary, nocase, nocasepref, noaccents, espdict, espnocs, espnoac

Changes to functions, commands, and stored procedures

Following are the changes to functions, commands, and stored procedures for Adaptive Server version 12.5.2.

The Real Time Meassages Sevices and the Web Services options add stored procedures as well. See the *Real Time Data Service User's Guide* and the *Web Services User's Guide* for more information.

Changed commands

union

The maximum number of subqueries within a single side of a union has increased from 16 to 50.

dbcc checkcatalog

dbcc checkcatalog performs sysindexes consistency checking and adds the fix parameters to fix any errors, if any. The partial syntax is:

dbcc checkcatalog [(*database_name*[, fix])

where fix determines whether dbcc fixes the sysindexes errors it finds. The default mode for checkcatalog is to not fix the errors. You must put the database into singleuser mode to use the fix option. The new sysindexes checks may result in new errors, not raised by dbcc checkcatalog in a previous versions of the server.

kill

Adaptive Server version 12.5.2 adds the statusonly parameter to the kill command. kill ...statusonly reports on the progress of a server process ID (spid) in rollback status. It does not terminate the spid. The statusonly report displays the percent of rollback completed and the estimated length of time in seconds before the rollback completes. The syntax is:

kill *spid* with statusonly

Where *spid* is the number of the process you are terminating.

For example, the following reports on the process of the rollback of spid number 13:

```
kill 13 with statusonly
spid: 13 Transaction rollback in progress. Estimated rollback completion: 17%
Estimated time left: 13 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 29%
Estimated time left: 9 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 40%
Estimated time left: 8 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 47%
Estimated time left: 7 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 55%
Estimated time left: 6 seconds
```

```
spid: 13 Transaction rollback in progress. Estimated rollback completion: 65%
Estimated time left: 5 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 73%
Estimated time left: 4 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 76%
Estimated time left: 3 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 83%
Estimated time left: 2 seconds
spid: 13 Transaction rollback in progress. Estimated rollback completion: 94%
Estimated time left: 0 seconds
```

If the rollback of the spid has completed when you issue kill...statusonly or if Adaptive Server is not rolling back the specified spid, kill...statusonly returns the following message:

```
Status report cannot be obtained. KILL spid:nn is not
in progress.
```

Additional commenting for *select **

Descriptions for queries using select * are fully expanded to describe the columns selected. For example:

```
create procedure myproc as select * from authors
```

Produces the following in syscomments:

```
 create procedure myproc as
--Adaptive Server has expanded all '*' elements in the following statement
select authors.au_id, authors.au_lname, authors.au_fname, authors.phone, autho
rs.address, authors.city, authors.state, authors.country, authors.posta
lco
```

Changes to *grant* and *revoke*

This sections describes changes to the grant and revoke commands for Adaptive Server version 12.5.2.

grant and set proxy issue warning for fipsflagger

grant dbcc and set proxy issue the following warning when they are issued while set fipsflagger option is enabled:

```
SQL statement on line number 1 contains Non-ANSI text.
The error is caused due to the use of DBCC.
```

Granting default permissions to system tables and stored procedures

Neither installmaster or installmodel for Adaptive Server version 12.5.2 grant default permissions for some system tables (listed below). Instead, the default permissions on the system tables are assigned when Adaptive Server builds a new database.

Adaptive Server version 12.5.2 adds the default permissions parameter to the grant and revoke commands, which allows you to grant and revoke the default permissions for the system tables listed below. The partial syntax is:

```
grant default permissions on system tables
revoke default permissions on system tables
```

where default permissions on system tables specifies that you grant or revoke the default permissions for the following system tables when you issue it from any database:

- sysalternates
- syscolumns
- syscomments
- sysdepends
- sysindexes
- syskeys
- sysobjects
- sysprocedures
- sysprotects
- syssegments
- systypes
- sysusers
- syslogs
- sysconstraints
- sysreferences
- sysusermessages
- sysattributes
- systabstats
- sysxtypes

- sysjars
- systhresholds
- syspartitions
- sysstatistics
- sysqueryplans

Default permissions on system tables also makes the following changes:

- Revokes sysobjects(audflags) permissions from public
- Grants permissions for sysobjects to sso_role

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

- sysdatabases
- sysdevices
- syslocks
- sysmessages
- sysprocesses
- systransactions
- sysusages
- sysconfigures
- syscurconfigs
- syslanguages
- syscharsets
- sysservers
- systimeranges
- sysresourcelimits
- syslogins
- sysremotelogins

The command also makes the following changes:

- Revokes select on sysdatabases(audflags) from public
- Revokes select on sysdatabases(deftabaud) from public

- Revokes select on sysdatabases(defvwaud) from public
- Revokes select on sysdatabases(defpraud) from public
- Grants select on sysdatabases to sso_role
- Revokes select on syslogins(password) to public
- Revokes select on syslogins(audflags) from public
- Grants select on syslogins to sso_role

Granting and revoking permissions for *update statistics*, *delete statistics*, and *truncate table*

Adaptive Server version 12.5.2 allows you to grant and revoke permissions for users, roles, and groups for the update statistics, delete statistics, and truncate table commands. Table owners can also provide permissions through an implicit grant by adding update statistics, delete statistics, and truncate table to a stored procedure and then granting execute permissions on that procedure to a user or role.

You cannot grant or revoke permissions for update statistics at the column level. You must have the sso_role to run update statistics or delete statistics on sysroles, syssrvroles, and sysloginroles security tables.

By default, users with the sa_role have permission to run update statistics and delete statistics on system tables other then sysroles, syssrvroles and sysloginroles, and can transfer this privilege to other users.

The partial syntax for grant and revoke is:

grant [truncate table | update statistics | delete statistics] on *table_name* to {*user_name* | *role_name*}

revoke [truncate table | update statistics | delete statistics] on *table_name* from {*user_name* | *role_name*}

You can also issue grant all to grant permissions on update statistics, delete statistics, and truncate table.

For example, the following allows user “harry” to use truncate table and updates statistics on the authors table:

```
grant truncate table on authors to harry
grant update statistics on authors to harry
```

The following revokes truncate table and update statistics privileges from “harry” on the authors table:

```
revoke truncate table on authors from harry
revoke update statistics on authors from harry
```

The following allows user “billy” to use the delete statistics command on the authors table:

```
grant delete statistics on authors to billy
```

The following revokes the delete statistics privileges from user “billy” on the authors table:

```
revoke delete statistics on authors from billy
```

The following grants truncate table and update and delete statistics privileges to all users with the oper_role (if users “billy” and “harry” possess the oper_role, they can now run these commands on authors):

```
grant truncate table on authors to oper_role
grant update statistics on authors to oper_role
grant delete statistics on authors to oper_role
```

The following revokes truncate table and update and delete statistics privileges from all users with the oper_role:

```
revoke truncate table on authors from oper_role
revoke update statistics on authors from oper_role
revoke delete statistics on authors from oper_role
```

Users “billy” and “harry” can no longer run these commands on authors.

You can also implicitly grant permissions for truncate table, delete statistics, and update statistics through a stored procedure. For example, assuming “billy” owns the authors table, he can run the following to grant “harry” privileges to run truncate table and update statistics on authors:

```
create procedure sproc1
as
truncate table authors
update statistics authors
go
grant execute on sproc1 to harry
go
```

You can also implicitly grant permissions at the column level for update statistics and delete statistics through stored procedures.

Note Once you grant permission to execute update statistics to a user, they also have permission to execute variations of this command, such as update all statistics, update partition statistics, update index statistics, update statistics *table*, and so on. For example, the following grants “billy” permission to run all variations of update statistics on the authors table:

```
grant update statistics on authors to billy
```

If you revoke a user's permission to execute update statistics, you also revoke their ability to execute the variations of this command.

You cannot grant variants of update statistics (for example, update index statistics) separately. That is, you *cannot* issue:

```
grant update all statistics to harry
```

However, you can write stored procedures that control who executes these commands. For example, the following grants "billy" execute permission for update index statistics on the authors table:

```
create proc sp_ups as
update index statistics on authors
go
revoke update statistics on authors from billy
go
grant execute on sp_ups to billy
```

You cannot grant and revoke delete statistics permissions at the column level.

Although Adaptive Server audits truncate table as a global, miscellaneous audit, it does not audit update statistics. To retain clear audit trails for both truncate table and update statistics, Sybase recommends that you include both commands in a stored procedure to which you grant users execute permission, as described above.

The command fails and generates error number 10330 **i**f a user issues update statistics, delete statistics or truncate table and they:

- Do not own the table.
- Do not have the sa_role.
- Are not a database owner who has successfully used setuser to become the user who is the owner of the table.
- Have not been granted update statistics, delete statistics, or truncate table privileges.

grant set proxy command

In earlier versions of Adaptive Server, set proxy allowed you to switch your server user identity to any other server login, but it did not allow you to limit use of set proxy based on the target login roles. When you granted set proxy to a user, this user could become any other server user.

As of Adaptive Server version 12.5.2, you can grant set proxy...restricted role, which allows you to restrict which roles cannot be acquired when switching identities.

The syntax for set proxy is:

grant set proxy to *user_or_role_list*
[restricted role *role_list* | all | system]

Parameters

- *user_or_role_list* – list of roles you are restricting for the target login. Both the grantee and target login must have all roles on this list or the command fails.
- *all* – ensures that all roles belonging to the grantee are granted to the target login.
- *system* – ensures the grantee has the same set of system roles as the target login.

Examples

- Example 1: This example grants set proxy to user "joe" but restricts him from switching identities to any user with the sa, sso, or admin roles (however, if he already has these roles, he can set proxy for any user with these roles):

```
grant set proxy to joe
restricted role sa_role, sso_role, admin_role
```

  When "joe" tries to switch his identity to a user with admin_role (in this example, Our_admin_role), the command fails unless he already has admin_role:

```
set proxy Our_admin_role
Msg 10368, Level 14, State 1:
Server 's', Line 2:Set session authorization
permission denied because the target login has a role
that you do not have and you have been restricted
from using.
```

  After "joe" is granted the admin_role and retries the command, it succeeds:

```
grant role admin_role to joe
set proxy Our_admin_role
```

- Example 2: Restricts "joe" from being granted any new roles when switching identities:

```
grant set proxy to joe
restricted role all
```

  "joe" can grant set proxy only to users who have the same (or lessor) roles than he has.

- Example 3: Restricts Joe from acquiring any new system roles when using set proxy:

```
grant set proxy to joe
restricted role system
```

  set proxy fails if the target login has system roles that joe lacks.

Usage

- You can restrict roles incrementally with grant set proxy. For example, you can first restrict the sa_role, then the sso_role:

```
grant set proxy to joe
restrict role sa_role
grant set proxy to joe
restrict role sso role
```

- You cannot unrestrict individual roles. You must revoke set proxy, to revoke permissions from all roles, as illustrated in this query:

```
select distinct user_name(p.uid), b.name, p.grantor,
                 Restricted_role=case
convert(tinyint,substring(isnull(p.columns,0x1),1,1)) & 1
                        when 1 then
                             "None"
                        else
                             isnull(role_name(c.number - 1), "System
"+convert(char,c.number))
                                      end
    from sysprotects p, master.dbo.spt_values b, master.dbo.spt_values c
    where
          convert(tinyint,substring(isnull(p.columns,0x1), c.low,1)) &
c.high = 0
         and c.type = "P" and c.number <= 1024 and c.number >0 and
p.action = 167
         and b.type = "T"
         and b.number = (p.protecttype + 204)
          and role_name(c.number - 1) is not null
```

optdiag

optdiag displays the statistic, sampling percent last used, which indicates that statistics are gathered with a user-specified sampling percent.

Changes to *update* and *delete*

In pre-12.5.2 versions of Adaptive Server, queries that used update and delete on views with a union all clause were sometimes resolved without using worktables, which occasionally lead to incorrect results. In Adaptive Server 12.5.2, queries that use update and delete on views with a union all clause are always resolved using worktables in tempdb. However, this behavior may result in a performance degradation.

Additions to *dbcc complete_xact*

dbcc complete_xact enables a System Administrator to commit or roll back a distributed transaction in circumstances where the external transaction coordinator cannot. In earlier versions of Adaptive Server, a transaction could not heuristically complete unless it was in the "prepare" state, and the transaction coordinator used a two-phase commit protocol to commit the transaction. However, in some cases, a transaction coordinator may want to use a one-phase commit protocol as an optimization to commit the transaction.

Adaptive Server 12.5.2 includes the 1pc parameter to the dbcc complete_xact command. 1pc heuristically completes a transaction that was subject to a one-phase commit protocol optimization—instead of the regular two-phase commit protocol—by the external transaction manager that was coordinating its completion. Heuristically committing such a transaction requires that the transaction is in a "done" state (as reported by sp_transactions).

The partial syntax for dbcc complete_xact is:

```
dbcc complete_xact("<xid>", "commit", "1pc")
```

Note Before heuristically committing the transaction, the System Administrator should make every effort to determine whether the coordinating transaction manager committed the distributed transaction.

The following example uses sp_transactions to determine the name of a one-phase commit transaction that did not heuristically commit because it was not in a "prepared" state. The example then explains how to use the 1pc parameter to successfully commit the transaction:

```
sp_transactions
xactkey                          type     coordinator     starttime
state            connection      dbid       spid      loid
failover        srvnname      namelen
xactname
------------------------------  --------- -----------    --------------------
```

```
--------------- ----------      ------      ------    -----------
----------      -----------   -----------------
-------------
0xbc0500000b00000030c316480100 External       XA          Feb  2 2004  1:07PM
Done-Detached     Detached         1         0        2099
Resident Tx     NULL             88
28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU_00003M00

00Y_:SYBBEV0A_LRM
(1 row affected)

(return status = 0)
```

If you try to commit this transaction:

```
dbcc
complete_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU
_00003M0000Y_:SYBBEV0A_LRM", "commit"))
```

Adaptive Server issues the following error message:

```
Msg 3947, Level 16, State 1:
Server 'PISSARRO_1251_P', Line 1:
A heuristic completion related operation failed. Please
see errorlog for more details.
DBCC execution completed. If DBCC printed error
messages, contact a user with System Administrator (SA)
role.
```

Because the transaction is in a "done" state, you can use a one-phase commit protocol optimization to heuristically complete the transaction after verifying the transaction was committed. You can commit this transaction using the dbcc complete_xact("1pc") parameter:

```
dbcc
complete_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM000FU
_00003M0000Y_:SYBBEV0A_LRM", "commit", "1pc")
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

You can remove the transaction from systransactions with the dbcc forget_xact command:

```
dbcc forget_xact("28_u7dAc31Wc380000000000000000000000000000000001HFpfSxkDM0
00FU_00003M0000Y_:SYBBEV0A_LRM")
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

If you run sp_transactions again, the previous transaction does not appear:

```
sp_transactions
```

```
xactkey                          type     coordinator     starttime
state            connection      dbid       spid      loid
failover        srvnname      namelen
xactname
------------------------------  --------- -----------    --------------------
--------------- ----------      ------      ------    -----------
-------------
(0 row affected)
```

Changes to *create procedure* (SQLJ)

Earlier versions of Adaptive Server did not allow you to define a default value for a SQLJ procedure parameter. With Adaptive Server 12.5.2, you can define a default value so that you can execute the SQLJ procedure without providing a parameter value.

This new capability of create procedure (SQLJ) mimics the existing behavior of create procedure.

Syntax

```
create procedure [owner.]sql_procedure_name
    ( [ [ in | out | inout ] sql_parameter_name
        sql_datatype [( length )] | (precision[, scale]) ]
        [=default]
    [ , [ in | out | inout ] sql_parameter_name
        sql_datatype [( length )] | (precision[, scale]) ]
         [=default] ... ] )
...
```

New option

default

defines a default value for the procedure's parameter. If a default is defined, you can execute the procedure without a parameter value. The default must be a constant. It can include the wildcard characters (%, _, [ ], and ^) if the procedure uses the parameter name with the keyword like.

The default can be NULL. The procedure definition can specify that some action be taken if the parameter value is NULL.

Examples

This procedure returns values that are always larger than 10:

```
create procedure my_max(a int = 10, b int = 10)
language java parameter style java
external name 'java.lang.Math.ma'

exec my_max
(return status = 10)

exec my_max 8
(return status = 10)
```

See also the examples for Transact-SQL create procedure.

Changes to stored procedures

Stored procedures changed for security enhancements

These stored procedures have additional security functionality:

- sp_modifylogin
- sp_displaylogin
- sp_addlogin

For more information, see "Enhanced login controls" on page 59.

sp_monitor

Earlier versions of Adaptive Server did not include parameters with sp_monitor. Adaptive Server version 12.5.2 adds the parameters described in this section.

Note Before using the new parameters associated with sp_monitor, you must set up monitoring tables and the related stored procedures needed to enable these options, which are part of the *installmontables* script. For more information, see "Installing Monitoring Tables" in *Performance and Tuning: Monitoring and Analyzing*.

Description

Displays statistics about Adaptive Server.

Syntax

sp_monitor [connection, [cpu | diskio | elapsed time]] [event, [*spid*]] [*procedure*, [*dbname*, [*procname*, [, summary | detail]]]] [enable] [disable] [statement, [cpu | diskio | elapsed time]] [help], [connection | statement | procedure | event]]

Parameters

connection

displays information on each connection. connection uses the following monitoring tables:

- monProcessSQLText
- monProcessActivity

cpu | diskio | elapsed time

these parameters order the output of sp_monitor connection. cpu indicates the amount of CPU time consumed by each different connection. diskio indicates the number of physical reads performed by each connection. elapsed time indicates the sum of the CPU time and the wait times for each connection.

event

displays information about the events each task spent time waiting for, and the duration of the wait. This is reported in descending order of wait time in milliseconds. event uses the following monitoring tables:

- monProcessWaits
- monWaitEventInfo

spid

allows you to obtain event information for a specific task by entering its *spid*. You must specify the numeric value of *spid* within quotation marks.

procedure

displays statistics about stored procedures:

- ProcName – the stored procedure being monitored.
- DBNAME – the database in which the stored procedure is located.
- NumExecs – the approximate number of executions of this specific stored procedure.
- AvgCPUTime – the average CPU time that it takes for the stored procedure to execute.
- AvgPhysicalReads – the average number of disk reads performed by the stored procedure.
- AvgLogicalReads – the average number of logical reads performed by the stored procedure.
- AvgMemUsed_KB – the average amount of memory in KB used by the stored procedure.

procedure uses the monSysStatement monitoring table.

dbname

displays information on procedures for the specified database.

procname

displays information on the specified procedure.

summary | detail

displays either summary information, which provides an average of all instances of the procedure, or detailed information, which provides information on every instance of the stored procedure.

enable

enables the new options for sp_monitor. It turns on the configuration parameter required to begin monitoring.

disable

disables monitoring.

statement

sp_monitor statement displays information on each statement. statement uses the following monitoring tables:

- monProcessSQLText
- monProcessStatement

cpu | diskio | elapsed time

These parameters help order the output of sp_monitor statement. cpu indicates the the cpu time consumed by different statements. diskio indicates the number of physical reads done by different statements. elapsed time indicates the sum of the CPU time and the wait times for different statements.

help

displays the syntax and examples for sp_monitor.

Examples

Example 1 This example shows how to display information about connections:

```
1> sp_monitor "connection"
2> go
spid    LoginName    ElapsedTime  LocksHeld  SQLText
----    ---------    -----------  ---------  -----------------
12      sa           90300        2          exec get_employee_salaries
27      sa           17700        1          exec get_employee_perks
```

By default, the output by default is sorted in the descending order of the ElapsedTime.

Example 2 This example identifies the connections performing the most physical reads:

```
1> sp_monitor "connection","diskio"
2> go
spid  LoginName  Physical_Reads  LocksHeld  SQLText
```

```
----  ---------  --------------  ---------  --------------------------
12    sa         117             2          exec get_employee_salaries
27    sa         1               0          exec get_employee_perks
```

Example 3 This example displays information about each statement:

```
1> sp_monitor "statement"
2> go
spid   LoginName   ElapsedTime   SQLText
----   ---------   -----------   --------------------------
12     sa          100           exec get_employee_salaries
```

Example 4 This example displays the events each task spent time waiting for and the duration of the wait, reported in descending order of wait times:

```
1> sp_monitor "event"
2> go
SPID   WaitTime    Description
------ ----------- ----------------------------------------
6      108200      hk: pause for some time
29     108200      waiting for incoming network data
10     107800      waiting while allocating new client socket
15     17100       waiting for network send to complete
14     5900        waiting for CTLIB event to complete
14     400         waiting for disk write to complete
7      200         hk: pause for some time
7      100         waiting on run queue after yield
12     100         waiting for network send to complete
```

Example 5 This example displays event data for spid 14:

```
1> sp_monitor "event","14"
2> go
WaitTime    Description
----------- ---------------------------------
9000 waiting for CTLIB event to complete
600 waiting for disk write to complete
200 waiting for disk write to complete
100 waiting on run queue after yield
100 wait for buffer write to complete
```

Example 6 This example provides a summary of most recently run procedures, sorted in descending order of average elapsed time. This example provides historical monitoring information rather than the current state.

```
1> sp_monitor "procedure"
2> go

Average Procedure Statistics
============================

ProcName       DBName    AvgElapsedTime   AvgCPUTime  AvgWaitTime
AvgPhysicalReads AvgLogicalReads AvgPacketsSent NumExecs
-----------------------------------------------------------------------
neworder_remote   tpcc   1833     16    1083     26   96    0     6
neworder_local    tpcc   1394     13    1181     31   122   0     38
tc_startup        tpcc   1220     3     1157     0    3     0     59
delivery          tpcc   1000     0     800      23   49    0     2
```

Usage

- You must run sp_monitor when a representative workload is running on the system.
- Typically, you will run procedures in this sequence:
  - Run sp_monitor enable
  - Invoke sp_monitor options
  - Run sp_monitor disable when you have completed the monitoring
- When you are using sp_monitor procedure, the number of rows returned can be very large; you may want to use the summary option instead of the detail option. It may also take a while for this command to complete on an active system.

Permissions

You must have mon_role permissions to execute sp_monitor. For more information see “Monitoring Tables” *Performance and Tuning: Monitoring and Analyzing*.

force parameter for *sp_dropalias*

sp_dropalias includes the optional parameter force, which allows you to drop an alias even if it owns database objects. Earlier versions of sp_dropalias required that you first drop the objects owned by the alias before dropping the alias.

For example, if you use the force parameter, you can drop the alias “harry,” which owns a procedure namelist. Adaptive Server drops the alias but issues a warning message:

```
sp_dropalias harry, force
Warning: You have forced the drop of the alias for login
```

```
'harry' which owns objects in the database. This may
result in errors when those objects are accessed from
or contain references to another database.
Alias user dropped.

(return status = 0)
```

sp_audit

sp_audit allows you to audit extended stored procedures.

sp_helptext

sp_helptext truncates trailing spaces when displaying the source text from syscomments

New functions, commands, and stored procedures

Following are the functions, command, and stored procedures added for Adaptive Server version 12.5.2.

audit_event_name

Returns a description of an audit event.

Syntax

audit_event_name(*event_id*)

Parameters

event_id

is the number of an audit event.

Examples

Example 1

Queries the audit trail for table creation events.

```
select * from audit_data where audit_event_name(event)
      = "Create Table"
```

Example 2

Obtains current audit event values. See Table 12-3 for a complete list of audit values and their descriptions.

```
create table #tmp(event_id int, description
      varchar(255))
go
declare @a int
select @a=1
while (@a<120)
begin
      insert #tmp values (@a, audit_event_name(@a))
      select @a=@a + 1
end
select * from #tmp
go
----------------------------------------
event_id    description
----------    -------------------
      1 Ad hoc Audit Record
      2 Alter Database
      ...
      104 Create Index
      105 Drop Index
```

Usage

Table 12-3 lists the ID and name of each of the 105 audit events.

Table 12-3: Audit events

Audit number and description		
1 Ad hoc audit record	36 Fatal error	71 Update view
2 Alter database	37 Nonfatal error	72 NULL
3 Alter table	38 Execution of stored procedure	73 Auditing enabled
4 BCP in	39 Execution of trigger	74 Auditing disabled
5 NULL	40 Grant command	75 NULL
6 Bind default	41 Insert table	76 SSO changed password
7 Bind message	42 Insert view	77 Table change
8 Bind rule	43 Load database	78 Audit option change
9 Create database	44 Load transaction	79 NULL
10 Create table	45 Log in	80 Role check performed
11 Create procedure	46 Log out	81 DBCC command
12 Create trigger	47 Revoke command	82 Config
13 Create rule	48 RPC in	83 Online database
14 Create default	49 RPC out	84 Setuser command
15 Create message	50 Server boot	85 UDR command

Audit number and description		
16 Create view	51 Server shutdown	86 Builtin function
17 Access to database	52 Curread modification	87 Disk release
18 Delete table	53 Curwrite modification	88 Set SSA command
19 Delete view	54 Update mode modification	89 Kill/terminate command
20 Disk init	55 Role toggling	90 Connect command
21 Disk refit	56 NULL	91 Reference
22 Disk reinit	57 NULL	92 Command text
23 Disk mirror	58 Truncation of audit table	93 JCS install command
24 Disk unmirror	59 NULL	94 JCS remove command
25 Disk remirror	60 NULL	95 Unlock admin account
26 Drop database	61 Access to audit table	96 Quiesce database command
27 Drop table	62 Select table	97 Create SQLJ function
28 Drop procedure	63 Select view	98 Drop SQLJ function
29 Drop trigger	64 Truncate table	99 SSL administration
30 Drop rule	65 Trusted procedure execution	100 Disk resize
31 Drop default	66 Trusted trigger execution	101 Mount database
32 Drop message	67 Unbind default	102 Unmount database
33 Drop view	68 Unbind rule	103 Login command
34 Dump database	69 Unbind message	104 Create index
35 Dump transaction	70 Update table	105 Drop index

Standards ANSI SQL – compliance level: Transact-SQL extension.

Permissions Any user can execute audit_event_name.

See also select, sp_audit

sp_ldapadmin

Description

Creates an LDAP URL search string, lists an LDAP URL search string, or verifies an LDAP URL search string or login.

Syntax

```
sp_ldapadmin { set_primary_url, 'ldapurl' |
        set_secondary_url, { 'ldapurl' | null } |
        set_access_acct, account_distinguished_name, account_password
        set_dn_lookup_url, ldapurl
        list_urls | check_url, 'ldapurl' |
        check_login, 'login_name' }

ldapurl::=ldap://host:port/node/?attributes?base | one | sub?filter
```

Parameters

set_primary_url, '*ldapurl*'
: creates the specified search string *ldapurl*. Exactly one primary search string can be created.

set_secondary_url, { '*ldapurl*' | null }
: creates the specified secondary search string *ldapurl* or no secondary search string. Exactly one secondary search string can be created.

set_access_acct
: allows you to perform administrative searches using account, distinguished name, and password information.

account_distintuished_name
: distinguished name of the directory services account.

distintuished_name
: a distinguished name is the name that uniquely identitifies an entry at a specific level of the hierarchy and describes the path of names that trace the entry back to the root of the hierachical tree.

account_password
: password associated with the distinguished name.

set_dn_lookup_url
: specifies an LDAP URL in which to search for the distinguished name. Setting this URL causes Adaptive Server to use an alternative authentication algorithm.

ldapurl
: URL of the string in which you are searching for the distinguished name associated with a user. The default attribute name is *entrydn*

list_urls
: displays LDAP URL search strings.

check_url, '*ldapurl*'
verifies an LDAP URL search string. Can also verify the existence of a user account, but it does not authenticate the user.

check_login, *login_name*
verifies a user account for the existing LDAP URL search strings. It does not authenticate the user.

host
is the host name of the LDAP server.

port
is the port number of the LDAP server.

node
specifies the node in the object hierarchy at which to start the search.

attributes
is a list of attributes to return in the result set. Each LDAP server may support a different list of attributes.

base | one | sub
qualifies the search criteria. base specifies a search of the base node; one specifies a search of node and one sublevel below node; sub specifies a search of node and all node sublevels.

filter
specifies the attribute or attributes to be authenticated. The filter can be simple, such as "uid=*," or compound, such as "(uid=*)(ou=*group*)." The syntax is LDAP server dependent and uses a wildcard (*) to describe the login name.

Examples

Example 1 Creates an LDAP URL search string for the LDAP SunONE Directory Server.

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
    ou=People,dc=MyCompany,dc=com??sub?uid=*'
```

The search string identifies a directory server listening on host name "voyager," port number 389 (the default LDAP protocol port), the base node to begin the search is within organizational unit (ou) "People," and the domain is "MyCompany.com." It returns all attributes that match the filter uid=*. Adaptive Server replaces the wildcard with the Adaptive Server login name that is to be authenticated.

Example 2 Creates an LDAP URL search string defined in OpenLDAP 2.0.25 using the criteria described in Example 1.

```
sp_ldapadmin set_primary_url,'ldap://voyager:389/
```

```
dc=MyCompany,dc=com??sub?cn=*'
```

Example 3 Sets the secondary LDAP URL search string to null, indicating no failover and no secondary LDAP server.

```
sp_ldapadmin set_secondary_url, null
```

Example 4 Creates an LDAP URL search string with a compound filter.

```
sp_ldapadmin set_primary_url, 'ldap://voyager:389/
    ou=people,dc=siroe,dc=com??sub?(&(uid=*)
    (ou=accounting))
```

Example 5 Specify the access account for searches.

```
sp_ldapadmin set_access_acct, 'cn=admin, ou=People, dc=mycompany, dc=com',
'admin secret password'
```

Example 6 Specifies the URL to search for an account's distinguished name. In this example, entrydn specifies the attribute's name whose value is the distinguished name. It uses a wildcard search (`uid=*`) for the user ID.

```
sp_ldapadmin set_dn_lookup_url, 'ldap://myhost:389/ou=People,dc=
mycompany,dc=com?entrydn?sub?uid=*'
```

Example 7 The following uses the active directory feature as the LDAP directory server (available only on Adaptive Server version 12.5.2 or later). This example comprises a series of steps using the alternative authentication algorithm, and sets the account information needed to search for a user's distinguished names:

1 Set the account information:

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

2 Set the primary URL:

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

3 Set the distinguished name URL search:

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

4 Specify the account name with the client connection:

```
% isql -Umylogin -Pmypassword
```

Adaptive Server authenticates the account name against the distinguished name returned by the LDAP search (specified by the distinguished name URL search) and your client connection password when you log in to the server.

5 The *samaccountname* attribute name of the URL filter parameter is the account login for both Windows and Adaptive Server. The following search returns the *distinguishedName* parameter:

```
'CN=MyGivenname MySurname,CN=Users,DC=mycompany,DC=com'
```

This *distinguishedName* parameter is used to bind the LDAP server to authenticate the user.

Usage

- The LDAP vendor determines the syntax of the search string. In all cases, the search string specifies the attribute name that uniquely identifies the user in the form "*attribute=wildcard*" as in "cn=*."
- The first attribute in a compound filter must define the Relative Distinguished Name (for example, `...sub?(uid=*)(ou=group)`). Otherwise, the authentication fails.
- If you set the distinguished name and password for the set_access_acct parameter to NULL, set_access_account may bind anonymously, if the LDAP server allows this.
- The primary function of the account search string is to perform LDAP searches to locate and return a user's full distinguished name.
- If set_dn_lookup_url finds more than one match to its search, only the first one is used for authentication.
- The maximum length of the *account distinguished name* parameter is 255 characters.
- The maximum length of the *account password* parameter is 64 characters.
- If more than one attribute value is returned from a set_dn_lookup_url search, only the first one is used to attempt the authentication bind.
- When a search string is added, Adaptive Server verifies that it uses valid LDAP URL syntax and that it references an existing node. To ensure that the valid string returns expected values, carefully choose and verify the search string when configuring Adaptive Server.
- The secondary URL search string enables failover to another LDAP server. Adaptive Server uses the primary URL search string unless the LDAP Server is not active or the search string is invalid. In this event, Adaptive Server uses the secondary URL search string for authentication.

Permissions

Only the System Security Officer can execute sp_ldapadmin.

dbcc stackused

Description	Database consistency checker (dbcc) checks the logical and physical consistency of a database and provides statistics, planning, and repair functionality.
Syntax	dbcc stackused
Parameters	stackused reports the maximum amount of stack memory used since the server first started
Examples	`dbcc stackused`
Usage	• The amount of stack memory used in the past is only an indication of possible needs in the future. Adaptive Server may require more stack memory than it used in the past. Run dbcc statckused periodically to find your current stack memory usage.
Standards	SQL92 – Compliance level: Transact-SQL extension.
Permissions	Only the System Administrator can run dbcc stackused.

Index

Symbols

A

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X