SYBASE®

New Features Guide

Adaptive Server® Enterprise

15.0.2

DOCUMENT ID: DC00641-01-1502-01

LAST REVISED: June 2007

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 15.0.2. It discusses the new features included in this versions of Adaptive Server.

How to use this book

This book includes descriptions for all features added to the Adaptive Server releases 12.5.3, 12.5.3a, 12.5.4, 15.0. 1 and 15.0.2. Each part of the book includes the feature descriptions for one of these releases.

Related documents

The Adaptive Server Enterprise documentation set consists of:

- 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 products.

- *ASE Replicator User's Guide* – describes how to use the Adaptive Server Replicator 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 Component Integration Services to connect remote Sybase and non-Sybase databases.

- The configuration guide for your platform – provides instructions for configuration tasks.

- *Full-Text Search Specialty Data Store User's Guide* – describes how to use the Full-Text Search feature 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 Adaptive Server.
- *Java in Adaptive Server Enterprise* – describes how to install and use Java classes as datatypes, 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).
- *Messaging Service User's Guide* – describes how to use Real Time Messaging Services to integrate TIBCO Java Message Service and IBM WebSphere MQ messaging services with all Adaptive Server database applications.
- *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 Adaptive Server.
- *Performance and Tuning Guide* – is a series of four books that explains how to tune Adaptive Server for maximum performance:
  - *Basics* – documents 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 (regular size when viewed in PDF format).
- *Reference Manual* – is a series of four books that contains the following detailed Transact-SQL information:

  - *Building Blocks* – describes Transact-SQL™ datatypes, functions, global variables, expressions, identifiers and wildcards, and reserved words.
  - *Commands* – describes Transact-SQL commands.
  - *Procedures* – describes Transact-SQL system procedures, catalog stored procedures, system extended stored procedures, and dbcc stored procedures.
  - *Tables* – describes 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. Full-size available only in print version; a compact version is available in PDF format.
- *Transact-SQL User's Guide* – documents Transact-SQL, the Sybase-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 Failover to configure an Adaptive Server as a companion server in a high availability system.
- *Unified Agent and Agent Management Console* – describes the Unified Agent, which provides runtime services to manage, monitor, and control distributed Sybase resources.
- *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.
- *New Features Open Server 15.0 and SDK 15.0 for Microsoft Windows, Linux, and UNIX* – describes new features for Open Server and SDK version 15.0.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase 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 SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

  Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase 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 Click Certification Report.

3 In the Certification Report filter select a product, platform, and timeframe and then click Go.

4 Click a Certification Report title to display the report.

❖ **Finding the latest information on component certifications**

1 Point your Web browser to Availability and Certification Reports at http://certification.sybase.com/.

2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.

3 Select Search to display the availability and certification report for the selection.

❖ **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. If prompted, enter your MySybase user name and password.

3 Select a product.

4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the "Technical Support Contact" role to your MySybase profile.

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

Conventions

SQL is a free-form language. There are no rules about the number of words you can put on a line, or where you must break a line. For readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1: Font and syntax conventions for this manual

Element	Example
Command names, procedure names, utility names, and other keywords display in sans serif font.	select sp_configure
Database names and datatypes are in sans serif font.	master database
File names, variables, and path names are in italics.	*sql.ini* file *column_name* *$SYBASE/ASE* directory
Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italic in Courier font.	`select` *`column_name`* `from` *`table_name`* `where` *`search_conditions`*
Type parentheses as part of the command.	`compute` *`row_aggregate`* `(`*`column_name`*`)`
Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates "is defined as".	`::=`
Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces.	`{cash, check, credit}`
Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets.	`[cash | check | credit]`
The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command.	`cash, check, credit`
The pipe or vertical bar (|) means you may select only one of the options shown.	`cash | check | credit`

Element	Example
An ellipsis (...) means that you can *repeat* the last unit as many times as you like.	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  sp_dropdevice [*device_name*]

  For a command with more options:

  select *column_name*
  from *table_name*
  where *search_conditions*

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

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

  ```
  select * from publishers
  ```

- Examples of output from the computer appear as follows:

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

(3 rows affected)
```

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

Adaptive Server sensitivity to the case of database objects, such as table names, depends on the sort order installed on Adaptive Server. You can change case sensitivity for single-byte character sets by reconfiguring the Adaptive Server sort order. For more information, see the *System Administration Guide*.

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.

PART 1

New Features for Adaptive Server Version 15.0.2

Part 1 includes description for these features:

CHAPTER 1

Feature and platform compatibility

This chapter provides feature and platform compatibility over for Adaptive Server versions 15.0.2, 15.0.1, 12.5.4, 12.5.3a and 12.5.3.

Adaptive Server features and platforms

Table 1-1 shows feature availability for supported operating systems in Adaptive Server version 15.0.2. “Y” indicates the feature is supported for that platform.

Adaptive Server version 15.0.2 includes features from Adaptive Server releases 12.5.3, 12.5.3a, 12.5.4, and 15.0.1.

Table 1-1: Adaptive Server features for supported operating systems

Adaptive Server options	Solaris 32-bit	Solaris 64-bit	HP-UX PA Risc 64-bit	IBM AIX 64-bit	Linux x86 32-bit	Windows x86 32-bit	Linux on Power 64-bit	Linux Opteron 64-bit	Sol Opteron 64-bit	HP-UX Itanium 64-bit	Windows Opteron X64
Security and directory services	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Cybersafe Kerberos	Y	Y				Y					
Pluggable Authentication Module	Y	Y		Y	Y		Y	Y	Y	Y	
Fine grained access control	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LDAP server directory	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LDAP user authentication	Y	Y	Y	Y	Y	Y		Y	Y	Y	

Adaptive Server options	**Solaris 32-bit**	**Solaris 64-bit**	**HP-UX PA Risc 64-bit**	**IBM AIX 64-bit**	**Linux x86 32-bit**	**Windows x86 32-bit**	**Linux on Power 64-bit**	**Linux Opteron 64-bit**	**Sol Opteron 64-bit**	**HP-UX Itanium 64-bit**	**Windows Opteron X64**
Platform Native Kerberos	Y	Y									
Secure Sockets Layer	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y
MIT Kerberos	Y	Y	Y	Y	Y		Y	Y	Y	Y	
Encrypted columns	Y	Y	Y	Y	Y	Y		Y	Y	Y	
High availability	Y	Y	Y	Y	Y	Y				Y	
Partitions	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Enhanced Full-Text Search (EFTS)	Y	Y	Y	Y	Y	Y		Y			
Features included in base Adaptive Server											
Cross-platform dump and load	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Job Scheduler	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Native XML	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ASE Replicator	Y	Y	Y	Y	Y	Y					
IPv6	Y	Y	Y		Y	Y					
Java option	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Web Services	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Distributed Transaction Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Content management (external file support)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Archived database access	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

CHAPTER 2

Encrypted Columns

Overview

Adaptive Server version 15.0.2 includes these enhancements to encrypted columns:

- Protects data from administrator – you can protect keys and encrypted columns with your own password to ensure privacy of data against the power of the DBO or System Administrator.
- Maintains application transparency using key copies protected by login passwords. That is, you can create key copies and assign them to individual users. Users can encrypt their key copies using their login passwords. Once a key copy is associated with a login password, users do not have to supply the key encryption password when they access data encrypted with the key.

- Provides for key recovery – You can recover access to a key after losing a password. The key owner sets up a recovery key copy, which can later be used to reencrypt the key after losing the password.
- Returns a default value for users without decrypt permission – you can create or alter a table to allow select statements to return specified default values for users who do not have decrypt permission. This allows you to run existing applications and reports without generating a permission error, while keeping private data secure against unauthorized users. Reports generated by unauthorized users do not reveal the encrypted data.
- Restricts automatic decrypt permissions – when the restricted decrypt permission configuration parameter is enabled, the System Security Officer explicitly grants decrypt permission, restricting access to data. When restricted decrypt permission is enabled:
  - Table owners are not implicitly granted decrypt permission. The schema owner does not have automatic and implicit access to user data, even in systems that rely on the system encryption password to access the keys.
  - Only users with the sso_role can grant decrypt permission. with grant option is supported for decrypt permission.
  - Implicit access through ownership chains across view and tables or procedures and tables is restricted.
- Adds datatypes – you can encrypt these additional datatypes: date, time, datetime, smalldatetime, money, smallmoney, big int, unsigned big int, bit, unichar and univarchar.

Protecting data from the system administrator

In Adaptive Server 15.0.2 you can protect keys with a password and keep data protected from the power of the administrator. Once data is protected using keys with user defined passwords, to process encrypted columns, users must have:

- select, insert, update, or delete permission on the column, depending on the type of access.

- Decrypt permission on the encrypted columns used in the target list of a select statement and in query predicates. Users without decrypt permission on the column receive a permissions-related error unless the column is defined with a decrypt_default. See “Returning default values for encrypted columns” on page 28 for more information.
- The password used to encrypt the key if the user password is specified when creating or altering an encryption key. Users who do not supply the correct password receive an error when attempting to select, insert or update encrypted columns or when referencing columns in a where clause. You need not supply a password if the encryption key is protected by the system encryption password or a login password; Adaptive Server can find these passwords internally.

Key protection through passwords

Adaptive Server 15.0.2 introduces new syntax to protect access to a key through a user defined password. Management of keys and passwords can be restricted to the specialized role of the key custodian.

Basics of key management

Key management consists of creating, dropping, and modifying encryption keys, distributing passwords, and providing for key recovery in the event of a lost password.

Adaptive Server manages the security of keys by keeping keys encrypted when not in use. There are actually two keys between the user and the data: the column encryption key (CEK) and the key encryption key (KEK). The CEK encrypts data and users must have access to it before they can access the encrypted data, but for security reasons it cannot be stored on disk in an unencrypted form. Instead, Adaptive Server encrypts the CEK with a KEK when you create or alter an encryption key. The KEK is also used to decrypt the CEK before you can access decrypted data. The KEK is derived internally from the system encryption password, a user-specified password, or a login password, depending on how you specify the key’s encryption with the create and alter encryption key statements. CEKs are stored in encrypted form in sysencryptkeys.

Figure 2-1 describes the steps for creating and storing a column encryption key for a create encryption key statement. The KEK is derived from a password and the KEK and the raw CEK are fed into the encryption function to produce an encrypted CEK.

Figure 2-1: Steps to create an encryption key

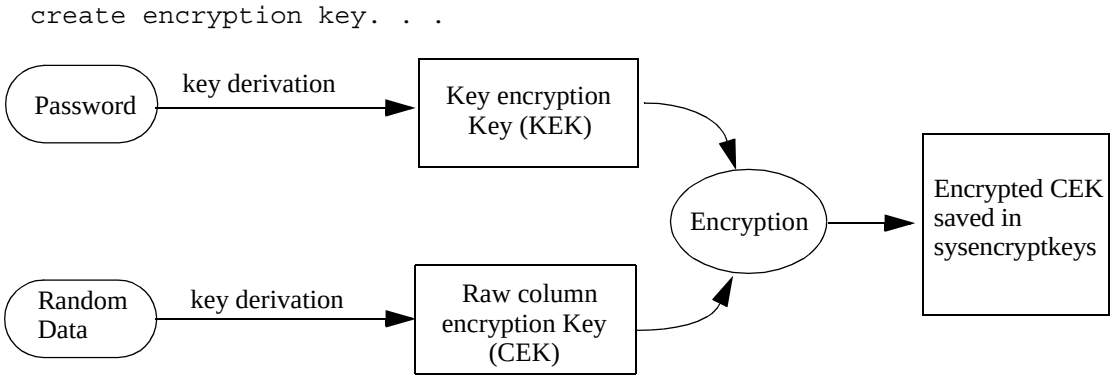
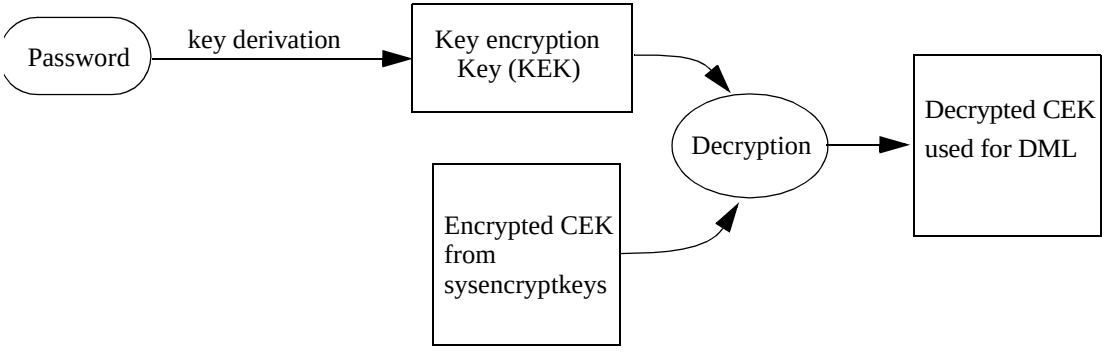


Figure 2-2 describes how the KEK is used during a DML operation to decrypt the CEK. The raw CEK is then used to encrypt or decrypt data.

Figure 2-2: Accessing a CEK to encrypt or decrypt on DML statement



The key custodian

The system administrator and DBO do not have implicit key management responsibilities. Instead of delegating all encryption responsibility to the SSO, Adaptive Server 15.0.2 adds system role keycustodian_role. The key custodian owns the encryption keys, but should have no explicit or implicit permissions on the data. The DBO grants users access to data through column permissions, and the key custodian allows them access to the key's password. keycustodian_role is automatically granted to sso_role and can be granted by a user with the sso_role.

The key custodian can:

- Create and alter encryption keys.
- Assign as the database default key a key he or she owns, as long as he or she also owns the current default key.
- Set up key copies for designated users, allowing each user access to the key through a chosen password or a login password.
- Share key encryption passwords with other users.
- Grant schema owners select access to encryption keys.
- Set the system encryption password.
- Recover encryption keys.
- Drop encryption keys they own.
- Change ownership of keys they own.

You can have multiple key custodians, who each own a set of keys. The key custodian grants the schema owner permission to use the keys on create table, alter table, and select into, and may disclose the key password to privileged users or allow users to associate key copies with a personal password or a login password. The key custodian can work with a "key recoverer" to recover keys in the event of a lost password or disaster (see "Key recovery commands" on page 26). If the key custodian leaves the company, the SSO can use the alter encryption key command to change key ownership to a new key custodian.

Key protection using system-encryption password

As in earlier versions, Adaptive Server 15.0.2 encrypts keys using the system encryption password by default. The SSO or key custodian sets the system encryption password using:

```
sp_encryption system_encr_passwd, 'password'
```

Using a system encryption password simplifies the administration of encrypted data because:

- Key management is restricted to setting up and changing the system encryption password.
- You need not specify passwords on create and alter encryption key statements.
- Password distribution and recovery from lost passwords are not required.
- Privacy of data is enforced through decrypt permission on the column. Restricted decrypt permission reinforces this privacy against the power of the administrator. See “Restricting decrypt permission” on page 36 for more information.

Key protection using user-specified passwords

You can limit the power of the SA or DBO to access private data when you specify passwords on keys through the create encryption key or alter encryption key statements. If keys have explicit passwords, before users can decrypt data, they need:

- Decrypt permission on the column
- The encryption key’s password

Users must also have knowledge of the password to run DML commands that encrypt data.

Use the create encryption key command to associate a password with a key:

```
create encryption key [db.[owner].]keyname  [as default]
[for algorithm_name] [with {[keylength num_bits]
[passwd 'password_phrase'] [init_vector {NULL | random}]
[pad {NULL | random}]}]
```

Where:

- for *algorithm_name* – (optional for the 15.0.2 release) specifies the algorithm you are using. The default is the Advanced Encryption Standard (AES) algorithm.
- *password_phrase* – is a quoted alphanumeric string of up to 255 bytes in length that Adaptive Server uses to generate the KEK.

Adaptive Server does not save the user-specified password. It saves a string of validating bytes known as the "salt" in sysencryptkeys.eksalt, which allows Adaptive Server to recognize whether a password used on a subsequent encryption or decryption operation is legitimate for a key. You must supply the password to Adaptive Server before you can access any column encrypted by keyname.

This example shows how to use passwords on keys, and the key custodian's function in setting up encryption. In this example the password on the key is shared among all users who have a business need to process encrypted data.

1 Key custodian Razi creates an encryption key:

```
create encryption key key1 with passwd 'Worlds1Biggest6Secret'
```

2 Razi distributes the password to all users who need access to encrypted data.

3 Each user enters the password before processing tables with encrypted columns:

```
set encryption passwd 'Worlds1Biggest6Secret' for key razi.key1
```

For more information about set encryption passwd, see "Accessing encrypted data" on page 11.

4 If the key is compromised because an unauthorized user gained access to the password, Razi alters the key to change the password.

Accessing encrypted data

You must supply the encryption key's password to encrypt or decrypt data on an insert, update, delete, select, alter table or select into statement. If the system encryption password protects the encryption key, you need not supply the system encryption password because Adaptive Server can already access it. Similarly, if your key copy is encrypted with your login password, Adaptive Server can access this password while you remain logged into the server (see "Application transparency using login passwords on key copies" on page 19). For keys encrypted with an explicit password, you must set the password in your session before executing any command that encrypts or decrypts an encrypted column with this syntax:

```
set encryption passwd 'password_phrase'
for {key | column} {key_name | column_name}
```

Where:

- *password_phrase* – is the explicit password specified with the create encryption key or alter encryption key command to protect the key.
- key – indicates that Adaptive Server uses this password to decrypt the key when accessing any column encrypted by the named key
- *key_name* – may be supplied as a fully qualified name. For example:

  ```
  [database_name.[owner_name].]key_name
  ```

- column – specifies that Adaptive Server uses this password only in the context of encrypting or decrypting the named column. End users do not necessarily know the name of the key that encrypts a given column.
- *column_name* – name of the column on which you are setting an encryption password. Supply *column_name* as:

  [database_name.[owner_name].] table_name.column_name

Each user who requires access to a key encrypted by an explicit password must supply the password. Adaptive Server saves the password for the user session's context in encrypted form. Adaptive Server removes the key from memory at the end of the session by overwriting the memory with zeros.

This example illustrates how Adaptive Server determines the password when it must encrypt or decrypt data. It assumes that the ssn column in the employee and payroll tables is encrypted with key1, as shown in these simplified schema creation statements:

```
create encryption key key1 with passwd "Ynot387"
create table employee (ssn char (11) encrypt with key1, ename char(50))
create table payroll (ssn char(11) encrypt with key1, base_salary float)
```

1. The key custodian shares the password required to access employee.ssn with Susan. He does not need to disclose the name of the key to do this.
2. If Susan has select and decrypt permission on employee, she can select employee data using the password given to her for employee.ssn:

```
set encryption passwd "Ynot387" for column employee.ssn
select ename from employee where ssn = '111-22-3456'
ename
-----------------------
Priscilla Kramnik
```

3. If Susan attempts to select data from payroll without specifying the password for payroll.ssn, the following select fails (even if Susan has select and decrypt permission on payroll):

```
select base_salary from payroll where ssn = '111-22-3456'
```

```
You cannot execute 'SELECT' command because the user
encryption password has not been set.
```

To avoid this error, Susan must first enter:

```
set encryption passwd "Ynot387" for column
payroll.ssn
```

The key custodian may choose to share passwords on a column-name basis and not on a key-name basis to avoid users hard coding key names in application code, which can make it difficult for the DBO to change the keys used to encrypt the data. However, if one key is used to encrypt several columns, it may be convenient to enter the password once. For example:

```
set encryption passwd "Ynot387" for key k1
select base_salary from payroll p, employee e
     where p.ssn = e.ssn
        and e.ename = "Priscilla Kramnik"
```

If one key is used to encrypt several columns and the user is setting a password for the column, they need to set password for all the columns they want to process. For example:

```
set encryption passwd 'Ynot387' for column payroll.ssn
set encryption passwd 'Ynot387' for column employee.ssn
select base_salary from payroll  p, employee e
where p.ssn = e.ssn
and e.ename = 'Priscilla Kramnik'
```

If a password is set for a column and then set at the key level for the key that encrypts the column, Adaptive Server discards the password associated with the column and retains the password at the key level. If two successive entries for the same key or column are entered, Adaptive Server retains only the latest. For example:

1 If a user miss-types the password for the column employee.ssn as “Unot387” instead of the correct “Ynot387”:

```
set encryption passwd "Unot387" for column
employee.snn
```

2 And then the user reenters the correct password, Adaptive Server retains only the second entry:

```
set encryption passwd "Ynot387" for column
employee.ssn
```

3 If the user now enters the same password at the key level, Adaptive Server retains only this last entry:

```
set encryption passwd "Ynot387" for key key1
```

4 If the user now enters the same password at the column level, Adaptive Server discards this entry because it already has this password at the key level:

```
set encryption passwd "Ynot387" for column
payroll.ssn
```

If a stored procedure or a trigger references a column encrypted by a user specified password, you must set the encryption password before executing the procedure or the statement that fires the trigger.

Note Sybase does not recommend that you place the set encryption passwd statement inside the trigger or procedure because this could lead to unintentional exposure of the password through sp_helptext. Additionally, hard-coded passwords require you to change the procedure or trigger when a password is changed.

Changing a key's password

You can use the alter encryption key command to change the current password for an encryption key using the following syntax. You can use this command on individual key copies. For more information about key copies, see "Accountability and application transparency through key copies" on page 16:

```
alter encryption key [database_name.[owner].] key_name
  [with passwd 'old_password' | system_encr_passwd | login_passwd]
  modify encryption
  [with passwd 'new_password' | system_encr_passwd | login_passwd]
```

Where:

- *key_name* – identifies a column encryption key.
- with passwd '*old_password*' – specifies the user-defined password previously specified to encrypt the base key or the key copy with a create encryption key or alter encryption key statement. The password can be up to 255 bytes long. If you do not specify with passwd, the default is the system encryption password.
- with passwd '*new_password*' – specifies the new password Adaptive Server uses to encrypt the column encryption key or key copy. The password can be up to 255 bytes long. If you do not specify with passwd, the default is system_encr_passwd.

- system_encr_passwd – the default encryption password Adaptive Server uses. You cannot modify the base key to encrypt it with the system encryption password if one or more key copies already exist. This restriction prevents the key custodian from inadvertently exposing an encryption key to access by an administrator after the key custodian has set up the key for restricted use by individual users. You cannot modify key copies to encrypt using the system encryption password.
- *login_passwd* – the login password of the current session. You cannot modify the base key to use *login_password* for encryption. A user can modify his own key copy to encrypt with his *login_password*. See "Application transparency using login passwords on key copies" on page 19 for alternatives to encrypt key copies with a user's login password that do not require the key copy assignee to execute the alter encryption key command.

In this example, the key custodian alters the base key because the password was compromised or a user who knew the password left the company.

1 Key custodian Razi creates an encryption key:

```
create encryption key key1 with passwd
'MotherOfSecrets'
```

2 Razi shares the password on the base key with Joe and Bill, who need to process the encrypted data (no key copies are involved).

3 Joe leaves the company.

4 Razi alters the password on the encryption key and then shares it with Bill, and Joe's replacement. They do not need to reencrypt any data because the underlying key has not changed, just the way the key is protected. The following statement decrypts key1 using the old password and reencrypts it with the new password:

```
alter encryption key key1
    with passwd 'MotherOfSecrets'
    modify encryption
    with passwd 'FatherOfSecrets'
```

Users, roles, and data access

User-specified passwords on encryption keys ensure that data privacy is protected from the powers of the System Administrator. Table 2-1 explains:

- How the key custodian can own the keys, but not see the data.

- How the DBO can own the schema, but not the data.
- How a user can see and process the data because of:
  - Key access, granted by the key custodian.
  - Data access, granted by the table owner.

Table 2-1: Permissions for users and roles on encrypted columns

Role	Can create encryption key?	Can use key in a schema definition?	Can decrypt encrypted data?
sso_role	Yes	No, requires create table permission	No. User with role may have knowledge of password, but requires select permission on table (SSO has implicit decrypt permission)
sa_role	No, requires create encryption key permission	Yes, but must be granted select permission on the key	No, requires knowledge of password
keycustodian_role. (See "The key custodian" on page 9 for more information)	Yes	No, requires create table permission	No. User with role may have knowledge of password, but requires decrypt and select permission on table or column.
DBO or schema owner	No, requires create encryption key permission	Yes, but must be granted select permission on the key	No, requires knowledge of password.
User	No	No	Yes, but must be granted decrypt or select permission and have knowledge of key's password.

Accountability and application transparency through key copies

Adaptive Server version 15.0.2 allows the key owner to create key copies, and this enables users to access encrypted columns using their copy of a single key. Key copies provide accountability for data because:

- A key copy is designated for an individual user with a private password known only to the user
- Adaptive Server does not save the passwords on disk.

- Without knowledge of the passwords protecting the key and its copies, not even the SA can access the data.

The use of keys with explicit passwords requires that Adaptive Server can access the password. To preserve application transparency, you may encrypt key copies with your login password, avoiding application changes to supply the key copy's password.

Creating key copies

Key copies are made available to individual users as follows:

- The key custodian uses create encryption key to create a key with a user-defined password. This key is known as the "base" key.
- The key custodian uses alter encryption key to assign a copy of the base key to an individual user with an individual password. See "Application transparency using login passwords on key copies" on page 19 for a description of for key copies accessed through a login password.

  This syntax shows how to add a key for a designated user encrypted using an explicit password:

  ```
  alter encryption key [ database_name.[ owner_name ].] key_name
      with passwd 'base_key_password'
      add encryption with passwd 'key_copy_password'
      for user ''
  ```

- *base_key_password* – is the password used to encrypt the base key, and may be known only by the key custodian. The password cannot be longer than 255 bytes. Adaptive Server uses the first password to decrypt the base column-encryption key.
- *key_copy_password* – the password used to encrypt the key copy. The password cannot be longer than 255 bytes. Adaptive Server makes a copy of the decrypted CEK, encrypts it with a KEK derived from the second *key_copy_password*, and saves the encrypted CEK copy as a new row in sysencryptkeys.
- *user_name* identifies the user for whom the key copy is made. For a given key, sysencryptkeys includes a row for each user who has a copy of the key, identified by their *user_id*.
- The key custodian adds as many key copies as there are users who require access through a private password.

- Users can alter their copy of the encryption key to encrypt it with a different password.

The following example illustrates how to set up and use key copies with an encrypted column:

1 Key custodian Razi creates the base encryption key with a user-specified password:

```
create encryption key key1 with passwd
'WorldsBiggestSecret'
```

2 Razi grants select permission on key1 to DBO for schema creation:

```
grant select on key key1 to dbo
```

3 DBO creates schema and grants table and column-level access to Bill:

```
create table employee (empname char(50), emp_salary money encrypt with
razi.key1, emp_address varchar(200))
grant select on employee to bill
grant decrypt on employee(emp_salary) to bill
```

4 Key custodian creates a key copy for Bill and gives Bill the password to his key copy. Only the key custodian and Bill know this password.

```
alter encryption key key1 with passwd 'WorldsBiggestSecret'
add encryption with passwd 'justforBill'
for user 'bill'
```

5 When Bill accesses employee.emp_salary, he first supplies his password:

```
set encryption passwd 'justforBill' for key Razi.key1
select empname, emp_salary from dbo.employee
```

When Adaptive Server accesses the key for the user, it looks up that user's key copy. If no copy exists for a given user, Adaptive Server assumes the user intends to access the base key.

Changing passwords on key copies

Once a user has been assigned a key copy, he or she can use alter encrypton key to modify the key copy's password. For the appropriate alter encryption key syntax, see "Changing a key's password" on page 14.

This example shows how a user assigned a key copy alters the copy to access data through his or her personal password:

- Key custodian Razi sets up a key copy on an existing key for Bill and encrypts it with a temporary password:

```
alter encryption key key1 with passwd 'MotherOfSecrets'
add encryption with passwd 'just4bill' for user bill
```

- Razi sends Bill his password for access to data through key1.
- Bill assigns a private password to his key copy:

```
alter encryption key Razi.key1 with passwd 'just4bill'
modify encryption with passwd 'billswifesname'
```

Only Bill can change the password on his key copy. When Bill enters the command above, Adaptive Server verifies that a key copy exists for Bill. If no key copy exists for Bill, Adaptive Server assumes you are attempting to modify the password on the base key and returns error message 10304: `Only the owner of object '<keyname>' or a user with sso_role can run this command`.

Application transparency using login passwords on key copies

The key custodian can set up key copies for encryption with a user's login password, providing:

- Ease of use. Users whose login password is associated with a key can access encrypted data without supplying a password.
- Better security. Users have fewer passwords to track, and are less likely to write them down.
- Lower administration overhead for key custodian. The key custodian need not manually distribute temporary passwords to each user who requires key access through a private password.
- Application transparency. Applications need not prompt for a password to process encrypted data. Existing applications can take advantage of the data privacy improvements in Adaptive Server version 15.0.2.

Use this syntax to encrypt a key copy with a user's login password:

alter encryption key [*database_name*.[*owner_name*].]*key_name*
with password '*base_key_password*'
add encryption for user '*user_name*' for login_association

Where login_association tells Adaptive Server to create a key copy for the named user, which it later encrypts with the user's login password. Encrypting a key copy with a login password requires two-steps.

1. Using the alter encryption key syntax above, the key custodian creates a key copy for each user who requires key access via a login password. Adaptive Server attaches information to the key copy to securely associate the key copy with a given user. The identifying information and key are temporarily encrypted using a key derived from the system encryption password. The key copy is saved in sysencryptkeys.
2. When a user processes a column requiring a key lookup, Adaptive Server notes that a copy of the encryption key identified for this user is ready for login password association. Using the system encryption password to decrypt the information in the key copy, Adaptive Server validates the user information associated with the key copy against the user's login credentials and encrypts the key copy with a KEK derived from the user's login password, which was supplied to the session.

When adding a key copy with alter encryption key, Adaptive Server initially encrypts the key copy with the system encryption password. The system encryption password must be available for Adaptive Server to decrypt the key copy when the user logs in. After Adaptive Server has reencrypted the key copy with the user's login password, the system encryption password is no longer required.

The following example encrypts a user's copy of the encryption key, key1, with the user's login password:

1. Key custodian Razi creates an encryption key:

```
create encryption key key1 for AES with passwd 'MotherofSecrets'
```

2. If there is not already a system encryption password, Razi sets one:

```
sp_encryption system_encr_passwd, 'keepitsecret'
```

3. Razi creates a copy of key1 for user Bill, initially encrypted with the system encryption password but eventually to be encrypted by Bill's login password:

```
alter encryption key key1 with
     passwd 'MotherofSecrets'
     add encryption
     for user 'bill'
     for login_association
```

4. Adaptive Server uses the system encryption password to encrypt a combination of the key and information identifying the key copy for Bill, and stores the result in sysencryptkeys.
5. Bill logs in to Adaptive Server and processes data, requiring the use of key1. For example, if emp.ssn is encrypted by key1:

```
select * from emp
```

Adaptive Server recognizes that it must encrypt Bill's copy of key1 with his login password. Adaptive Server uses the system encryption password to decrypt the key value data saved in step 3. It validates the information against the current login credentials, then encrypts key1's key value with a KEK generated from Bill's login password.

6 During future logins when Bill processes columns encrypted by key1, Adaptive Server accesses key1 directly by decrypting it with Bill's login password.

Users who are aliased to Bill cannot access the data encrypted by key1 because their own login passwords cannot decrypt key1.

7 When Bill loses authority to process confidential data, the key custodian drops Bill's access to the key:

```
alter encryption key key1
     drop encryption
     for user 'bill'
```

A user can encrypt a key copy with a login password with alter encryption key. However, the disadvantages of using this method over the login association are:

- The key custodian must communicate the key copies first assigned password to the user.
- The user must issue alter encryption key to reencrypt the key copy with a login password.

For example:

- Razi adds a key copy for user "bill" encrypted by an explicit password:

```
alter encryption key key1 with passwd 'MotherofSecrets"
add encryption with passwd "just4bill' for user bill
```

- Razi shares the key copy's password with "bill."
- "bill" decides to encrypt his key copy with his login password for his own convenience. He uses the command:

```
alter encryption key k1 with passwd "just4bill"
modify encryption with passwd login_passwd
```

- Nw, when "bill" processes encrypted columns, Adaptive Server accesses "bill's" key copy through his login password.

Accessing encrypted data using key copies

When a key custodian initially creates a key using create encryption key, the object that is created is defined as the "base key." The key custodian makes copies of the base key using the alter encryption key ... add encryption syntax. Users access a key through a key copy, if one was assigned to them, or through the base key. The base key is identical to its copies after decryption, only differing in the KEK used to encrypt and decrypt the key.

Access to the raw key is necessary:

- To create the key or alter it.
- To encrypt or decrypt data in an encrypted column.

The rules Adaptive Server follows for finding the correct key or key copy are:

- If you are the key owner – Adaptive Server accesses the base key. Typically, a key custodian has no authority or responsibility to process encrypted data so accessing the key on behalf of the key owner is confined to DDL operations on the key according to this algorithm:
  - Creating an encryption key – only the base key is affected
  - Altering an encryption key to add a key copy – Adaptive Server accesses the key by decrypting the base key. Adaptive Server copies, encrypts, and saves the key as a key copy.
  - Altering an encryption key to modify the password – Adaptive Server decrypts then re-encrypts the base key.
  - Altering an encryption key to drop a key copy – Adaptive Server does not require access through the base key, so it is not affected.
  - Dropping the encryption key – Adaptive Server does not require the raw key to drop the key and all its copies.
- If you are not the key owner – Adaptive Server finds a key copy assigned to you, and:
  - Your actions are restricted to modifying the password on your key copy. If you are not assigned a key copy, the alter encryption key statement fails unless you have the sso_role. For non-key owners who have sso_role, Adaptive Server verifies that the old password you have supplied is valid for the base key, and applies the new password to the base key.

- If your access to the key is for data manipulation language (DML) command, or a select statement that affects an encrypted column, Adaptive Server looks for a key copy assigned to you. If no such key copy exists, the server assumes the password you supplied is for the base key and Adaptive Server attempts to decrypt the base key with this password. If the key decryption is unsuccessful, you are denied access to the key and the data.
- Key copy access through a guest user is not allowed. For example, if a user is not a valid user in a database and the guest user has a key copy, this user is not allowed access to encrypted data through the guest user's key copy.

Login password change and key copies

If you hold a key copy encrypted by a login password on one or more keys, you need not modify the key copies after using sp_password to change your login password. As part of changing the login password, sp_password decrypts your key copies with your old login password and re-encrypts them with the new login password. For more information, see "sp_password" on page 61.

If the SSO uses sp_password to change your password without supplying your old password, sp_password drops your key copies. This prevents an administrator from gaining access to a key through a known password. After a mandatory password change of this kind, the key custodian must use alter encryption key to add key copies for login_association for the user whose password is changed. sp_password ignores offline databases and, for these keys, the key custodian follows the steps for recovering a lost key copy password when the database comes back online. See "Loss of login password" on page 25 for more information.

The key custodian may also need to perform these steps when a user's password is changed after starting the server using the -p flag. If the System Security Officer, who uses the -p flag, also has access to keys through key copies encrypted with his or her login password, then the key custodian must drop and recreate the SSO's key copies.

Dropping a key copy

When a user changes jobs or leaves the company, the key custodian drops this user's key copy with:

```
alter encryption key key_name
      drop encryption for user user_name
```

For example, if user Bill leaves the company, the key owner can prevent Bill's access to key1 by dropping Bill's key copy:

```
alter encryption key key1
       drop encryption for user Bill
```

Adaptive Server does not require a password for this command because no key decryption is required. This command removes Bill's key copy row, which drops Bills access to the key.

drop encryption key drops the base key and all its copies.

Recovering keys from lost passwords

This section describes how to recover key access from:

- Loss of a user-specified encryption password on a key copy.
- Loss of login password.
- Loss or unavailability of password on base key.

Loss of password on key copy

If the key custodian makes copies of the encryption key available to users and if a user loses his or her password, the key custodian drops the user's copy of the encryption key and issues another copy of the encryption key to the user with a new password.

In this example, the key custodian assigned a copy of key1 to Bill, and Bill changed his password on key1 to a password known only to him. After losing his password, Bill requests a new key copy from the key custodian.

1 The key custodian deletes Bill's copy of the key:

```
alter encryption key key1
drop encryption for user bill
```

2 The key custodian makes a new copy of key1 for user Bill and gives Bill the password:

```
alter encryption key key1
```

```
        with passwd 'MotherofSecrets'
        add encryption with passwd 'over2bill'
        for user bill
```

3 Bill automatically has permission to alter his own copy of key1:

```
alter encryption key key1
     with passwd 'over2bill'
     modify encryption
     with passwd 'billsnupasswd'
```

Loss of login password

If user Bill, who has key copies encrypted by his login password, loses his login password, you can recover his access to encryption keys with these steps:

1 The SSO issues Bill a new login password with sp_password. Adaptive Server drops any key copies assigned to Bill for login association or key copies already encrypted by Bill's login password.

2 The key custodian follows the regular procedures for setting up key encryption by login association, he verifies that the system encryption password was set, and creates Bill's key copy:

```
alter encryption key k1 with passwd 'masterofsecrets'
add encryption for bill for login_association
```

This step assumes the key custodian still knows the base key's password. If the key's encryption password is unknown, the key custodian must first follow the key recovery procedure. See "Loss of password on base key" on page 25 for more information.

3 The next time Bill accesses data encrypted by k1, Adaptive Server reencrypts Bill's key copy using Bill's new login password. For example, if emp_salary is encrypted by key k1, the following statement automatically alters Bill's key copy, reencrypting it with Bill's login password:

```
select emp_salary from emp where name like 'Prisicilla%'
```

Loss of password on base key

Key custodians can use key recovery if the base key password is lost. Key recovery is vital because, without the password, the key custodian cannot change the key's password or add key copies.

If all users share access to data through the base key and a user forgets the password, he or she can get the password from another user or the key custodian. If no one remembers the password, all access to the data is lost. Because of this, Adaptive Server recommends that keys be backed up by a recovery key copy.

The key custodian should:

1 Appoint one user as the key recoverer. The key recoverer's responsibility is to remember the password to the key recovery copy.

2 Make a copy of the base key for the key recoverer. Every key that requires recovery after a disaster must have a key recovery copy.

After an unforeseen circumstance, the key custodian may not be available. The steps for recovery of key1 are:

1 Change the key ownership to a new key custodian. See "Changing ownership of encryption keys" on page 28 for more information.

2 The key recoverer gives the password of the key copy to the new key owner, who recovers the CEK and assigns a new password to it.

Key recovery commands

The key recovery copy provides a way to recover a raw key if the password on the base key was lost or if the person who knows the password is unavailable. Adaptive Server does not allow access to data through the recovery key copy. A key recovery copy exists only to provide a backup for accessing the base key.

You can set up a recovery key copy with the following:

```
alter encryption key key_name with passwd base_key_passwd
add encryption with passwd recovery_passwd
for user key_recovery_user for recovery
```

Where:

- *base_key_passwd* – password the key custodian assigned to the base key in the create encryption key or alter encryption key command.
- *recovery_passwd* – password used to protect the key recovery copy.
- *key_recovery_user* – user assigned the responsibility for remembering a password for key recovery.

After setting the key recovery copy, the key custodian shares the password with the key recovery user, who can alter the password to one only the key recoverer knows:

alter encryption key *key_name* with passwd *old_recovery_passwd*
modify encryption with passwd *new_recovery_passwd* for recovery

The key recovery user keeps the password for the recovery key copy in a safe place until it is required to recover the base key.

During key recovery, the key recovery user tells the key custodian the password of the key recovery copy. The key custodian restores access to the base key through a known password with the following command:

alter encryption key *key_name* with passwd *recovery_key_passwd*
recover encryption with passwd *new_base_key_passwd*

Where:

- *recovery_key_passwd* – the password associated with the key recovery copy, shared with the key custodian by the recovery key user. Adaptive Server uses the *recovery_key_passwd* to decrypt the key recovery copy to access the raw key.
- *new_base_key_passwd* – the password used to encrypt the raw key. Adaptive Server updates the base key row in sysencryptkeys with the result.

After an unforeseen circumstance, it may be necessary to change ownership of the key to another key custodian. For more information, see “Changing ownership of encryption keys” on page 28.

The following example shows how to set up the recovery key copy and use it for key recovery after losing a password:

1 The key custodian originally creates a new encryption key protected by a password.

```
create encryption key key1 for AES passwd 'loseitl8ter'
```

2 The key custodian adds a special encryption key recovery copy for key1 for user “charlie.”

```
alter encryption key key1 with passwd 'loseitl8ter'
     add encryption
     with passwd 'temppasswd'
     for user charlie
     for recovery
```

3 “charlie” assigns a different password to the recovery copy and saves this password in a locked drawer:

```
alter encryption key key1
     with passwd 'temppasswd'
     modify encryption
     with passwd 'finditl8ter'
     for recovery
```

4 If the key custodian loses the password for base key, he can obtain the password from “charlie” and recover the base key from the recovery copy of the key using:

```
alter encryption key key1
     with passwd 'finditl8ter'
     recover encryption
     with passwd 'newpasswd'
```

The key custodian now shares access to key1 with other users by sharing the base key’s password or by dropping and adding key copies where changes in personnel have occurred.

Changing ownership of encryption keys

Changing ownership may occur in the normal course of business, or as part of key recovery following an unforeseen circumstance. This command, when executed by the SSO, transfers key ownership to a named user:

alter encryption key [[*database_name*].[*owner*].]*keyname*
modify owner *user_name*

Where user_name is the name of the user to be the new key owner. This user must be a user in the database where the key was created.

For example, if key owner “Razi” is the key custodian and owns the key encr_key, but is being replaced by a new key custodian named “tinnap,” use the following command to change the key ownership:

```
alter encryption key encr_key modify owner tinnap
```

Only the SSO or the key owner can run this command.

Returning default values for encrypted columns

This section describes how encrypted column uses default decrypt values.

Defining a decrypt default

The decrypt_default parameter for create table and alter table allows an encrypted column to return a user-defined value when a user without decrypt permission attempts to select information from the encrypted column. This avoids error message 10330:

```
Decrypt permission denied on object <table_name>,
database <database name>, owner <owner name>
```

Using decrypt defaults on encrypted columns allows existing reports to run to completion without error, and allows users to continue seeing the information that is not encrypted. For example, if the customer table contains the encrypted column creditcard, you can design the table schema so that:

```
select * from customer
```

Returns the value "**********" instead of returning the credit card data to users who lack decrypt permission.

Adding and removing a decrypt default

These examples show how to add and remove decrypt defaults from tables.

- Specify a decrypt default on a new column with create table. The partial syntax for create table is:

  create table *table_name* (*column_name datatype*
  [[encrypt [with *key_name*]] [decrypt_default *value*]], ...)

  Where:

  - decrypt_default – specifies that this column returns a default value for users who do not have decrypt permissions.
  - *value* – is the constant value Adaptive Server returns on select statements instead of the encrypted value. The value can be NULL on nullable columns only. If the decrypt_value cannot be converted to the column's data type, Adaptive Server catches the conversion error only when it executes the query.

  For example, the ssnum column for table t2 returns "?????????" when a user without decrypt permissions performs a query on it:

  ```
  create table t2 (ssnum char(11) encrypt
  decrypt_default '???????????', ...)
  ```

- The command to add a new encrypted column with a decrypt default is:

  alter table *table_name* add *column_name* type
  [[encrypt [with *key_name*] [decrypt_default *value*]], ...

- The syntax to add encryption and a decrypt default value to an existing column not previously encrypted, is:

alter table *table_name* modify *column_name* [type]
[[encrypt [with *key_name*]] [decrypt_default *value*]], …

For example, this modifies the emp table to encrypt the ssn column and specifies decrypt default:

```
alter table emp modify ssn encrypt with key1
decrypt_default '000-00-0000'
```

- To add a decrypt default to an existing encrypted column or change the decrypt default value on a column that already has a decrypt default using alter table.. replace. The syntax is:

  alter table *table_name* replace *column_name* decrypt_default *value*

  For example, this adds a decrypt default to the salary column, which is already encrypted:

  ```
  alter table employee replace salary decrypt_default
  $0.00
  ```

- To remove a decrypt default from an encrypted column without removing the encryption property, the syntax is:

  alter table *table_name* replace *column_name* drop decrypt_default

  For example, this removes the decrypt default for salary without removing the encryption property:

```
alter table employee replace salary drop decrypt_default
```

Permissions and decrypt default

You must grant decrypt permission on encrypted columns before users or roles can select or search on encrypted data they contain. If an encrypted column has a decrypt default attribute, users without decrypt permission can run queries that select or search on these columns, but the cleartext data does not display. (Adaptive Server does not issue the 10330 error message and does not decrypt the data).

In this example, the owner of table emp allows users with the hr_role to view emp.ssn. Because the ssn column has a decrypt default, users who have only select permission on emp and who do not have the hr_role see the decrypt_default value only and not the actual decrypted data.

```
grant select permission on table emp to public
grant decrypt on emp(ssn) to hr_role
```

If you have the hr_role and select from this table, you see the values for ssn:

```
select name, ssn from emp
name                          ssn
----------------------------  ------------
Joe Cool                      123-45-6789
Tinna Salt                    321-54-9879
```

If you do not have the hr_role and select from the table, you see the decrypt default:

```
select name, ssn from emp
name                              ssn
------------------------------    -----------
Joe Cool                          000-00-0000
Tinna Salt                        000-00-0000
```

order by clauses have no effect on the result set if you do not have the hr_role for this table.

Columns with decrypt default values

There are no restrictions on how you use columns with the decrypt default attribute in a query. You can use them in a target list expression, order by, group by, or subquery. Although expressions on the decrypt default constant value may not have a practical use, placing a decrypt default on a column does not impose any restrictions on use of the column in a Transact-SQL statement.

This example uses an expression on a column with a decrypt default value in the target list:

```
create table emp_benefits (col1 name char(30),
salary float encrypt decrypt_default -99.99)
```

If you perform a select against this table, but do not have decrypt permission, you see:

```
select salary/12 as monthly_salary from emp_benefits
where name = 'Bill Smith'
monthly_salary
---------------------
8.332500
```

When Adaptive Server returns a column's decrypt default value on a select into command, this decrypt default value is inserted into the target table. However the target column does not inherit the decrypt default property. You must use alter table to specify a decrypt default on the target table.

Decrypt default columns and query qualifications

If you use a column with the decrypt default property in a where clause, the qualification fails if you do not have decrypt permission. These examples use the emp table described above. Only users with the hr_role have decrypt permission on ssn.

1 If you have the hr_role and issue the following query, Adaptive Server returns one row.

```
select name from emp where ssn = '123-456-7890'
name

------------------------------
Joe Cool
```

2 If you do not have the hr_role, Adaptive Server returns no rows:

```
select name from emp where ssn = '123-456-7890'
name

------------------------------
(0 rows affected)
```

3 If you have the hr_role and include an or statement on a non-encrypted column, Adaptive Server returns the appropriate rows:

```
select name from emp where ssn = '123-456-7890' or
name like 'Tinna%'
name
------------------------------
Joe Cool
Tinna Salt
```

4 If you do not have the hr_role and issue the same command, Adaptive Server returns only one row:

```
select name from emp where ssn = '123-456-7890' or
name like 'Tinna%'
name
------------------------------
Tinna Salt
```

In this case, the qualification against the encrypted column with the decrypt default property fails but the qualification against the non-encrypted column succeeds.

If you do not have decrypt permission on an encrypted column, and you issue a group by statement on this column with a decrypt default, Adaptive Server groups by the decrypt default constant value.

decrypt default and implicit grants

If you do not have explicit or implicit permission on a table, Adaptive Server returns the decrypt default value.

In this example (using the emp table described above), the DBO creates the p_emp procedure which selects from the DBO-owned emp table:

```
create procedure p_emp as
     select name, ssn from emp
grant exec on p_emp to corp_role
```

Because you have the corp_role, you have implicit select and decrypt permission on emp

```
exec p_emp
name                           ssn
-----------------------------  ------------
Tinna Salt                     123-45-6789
Joe Cool                        321-54-9879
```

If the emp table and p_emp stored procedure were created by different users, you must have select permission on emp to avoid permissions errors. If you have select permission but not decrypt permission, Adaptive Server returns the decrypt default value of emp.ssn.

In this next example, "joe," a non-DBO user, creates the v_emp view, which selects from the DBO-owned emp table. In this case any permissions granted on the view are not implicitly applied to the base table.

```
create view v_emp as
     select name, ssn from emp
grant select on v_emp to emp_role
grant decrypt on v_emp to emp_role
```

Although you have the emp_role, when you issue:

```
select * from joe.v_emp
```

Adaptive Server returns the following because decrypt permission on dbo.emp.ssn has not been granted to the emp_role, and there is no implicit grant to emp_role on dbo.emp.ssn:

```
name                         ssn
--------------------------   ---------------
Tinna Salt                   000-00-0000
Joe Cool                     000-00-0000
```

decrypt default and *insert*, *update*, and *delete* statements

The decrypt default parameter does not affect target lists of insert and update statements.

If you use a column with a decrypt default value in the where clause of an update or delete statement, Adaptive Server may not update or delete any rows. For example, when using the emp table and permissions from the previous examples, if you do not have the hr_role and issue the following query, Adaptive Server does not delete the user's name:

```
delete emp where ssn = '123-45-6789'
(0 rows affected)
```

Decrypt default attributes may indirectly affect inserting and updating data if an application, particularly one with a graphical user interface (GUI) process, such as:

1 Selects data

2 Allows the user to update any of the data

3 Applies the row back to the same or a different table

If the user does not have decrypt permission on the encrypted columns, the application retrieves the decrypt default value and may automatically write the the unchanged decrypt default value back to the table. To avoid over-writing valid data with decrypt default values, you can use a check constraint to prevent these values from being automatically applied. For example:

```
create table customer (name char(30)),
cc_num int check (cc_num != -1)
encrypt decrypt_default -1
```

When the application selects data from the customer table, the user sees this data if he or she does not have decrypt permission on cc_num:

```
name                               cc_num
--------------------               ------------
Paul Jones                         -1
Mick Watts                         -1
```

However, if the user changes a name and updates the database, and the application attempts to update all fields from the values displayed, the default value for cc_num causes Adaptive Server to issue error 548:

```
"Check constraint violation occurred, dbname =
<dbname>, table name = <table_name>, constraint name =
<internal_constraint _name>"
```

This protects the integrity of the data. For a better solution, you can filter these updates when you write the application's logic.

Removing decrypt defaults

You can remove the decrypt default using any of these commands:

- drop table
- alter table .. modify .. drop col
- alter table .. modify .. decrypt
- alter table .. replace .. drop decrypt_default

For example, to remove the decrypt default attribute from the ssn column, enter:

```
alter table emp replace ssn drop decrypt_default
```

If you do not have the hr_role and select from the emp table after the table owner removed the decrypt default, Adaptive Server returns error message 10330.

Restricting decrypt permission

Adaptive Server version 15.0.2 enhances data privacy from the administrator even if you continue using the system encryption password for key protection. If you prefer to avoid password management and continue to use the system encryption password to protect encryption keys, you can restrict access to private data from the DBO by setting the restricted decrypt permission configuration parameter. This parameter grants the system security officer (SSO) control over which users have decrypt permissions. Once restricted decrypt permission is enabled, the SSO is the only user who receives implicit decrypt permission and who has implicit privilege to grant that permission to others. The SSO determines which users receive decrypt permission, or delegates this job to another user by granting decrypt permission with the with grant option. Table owners do not automatically have decrypt permission on their tables.

Users with execute permission on stored procedures or user-defined functions do not have implicit permission to decrypt data selected by the procedure or function. Users with decrypt permission on a view column do not have implicit permission to decrypt data selected by the view.

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

When run by the SSO, the following command enables or disables restricted decrypt permission in all databases:

```
sp_configure "restricted decrypt permission", [1 | 0]
```

When restricted decrypt permission is set to 0 (off), decrypt permission on encrypted columns acts the same as in earlier versions:

- The table owner or the SSO explicitly grants decrypt permission. However, with grant option on decrypt permission is supported.
- Decrypt permission is granted implicitly to table owners and the SSO.
- Decrypt permission is granted implicitly to any user through a chain of ownership. For example, if user Fred owns the proc1 stored procedure, which selects data from the encrypted column fred.table1.col1, and if Fred grants exec permission on proc1 to Harry, then Harry has implicit decrypt permission on fred.table1.col1.

- Decrypt permission is not needed for alter table decrypt. because the table owner has implicit decrypt permission on encrypted columns.

When restricted decrypt permission is set to 1 (on):

- Decrypt permission is granted implicitly to the SSO only.
- The SSO can grant decrypt permission using the with grant option parameter. This allows the SSO to decide who should grant decrypt permission in the system. For example, if the SSO wants user1 to be able to grant decrypt permission on user3.user3_tab, he or she issues:

  ```
  grant decrypt on user3.user3_tab to user1
  with grant option
  ```

  If you use a system encryption password, Sybase recommends that you do not grant decrypt permission to the DBO to protect data's privacy from the administrator. In a system that protects keys through user encryption passwords, you do not threaten the privacy of data by granting decrypt permission to the DBO with the with grant option. Access to keys through user passwords prevents the DBO and other parties from accessing the data unless they have a key's password; however, you may find it convenient for the DBO to decide which users should see the decrypted data. If you are not protecting keys and data with user-specified passwords, the System Security Officer should retain the sole responsibility to grant decrypt permission.
- Table ownership does not give a user implicit decrypt permission. That is, if you create a table with encrypted columns, you do not have decrypt permission on them unless this is explicitly granted to you.
- No user is implicitly granted decrypt permission through an ownership chain. For example, if user Fred owns the proc1 stored procedure, which selects data from the encrypted column fred.table1.col1, and if Fred grants exec permission on proc1 to Harry, then Harry must also have explicit decrypt permission on fred.table1.col1 to see the data.
- Aliased users assume the permissions of the user to whom they are aliased. Similarly, a user with sa_role, who is implicitly aliased to the DBO in any database, inherits any decrypt permissions explicitly granted to the DBO.
- Decrypt permission is required for alter table decrypt statement because the table owner does not have implicit decrypt permission on the table.

If you change restricted decrypt permission from 0 to 1, currently executing statements that use implicit decrypt permission complete. Any subsequent statements that use implicit decrypt permission fail with this error message until the SSO grants the user decrypt permission on the necessary columns:

```
Msg 10330 "DECRYPT permission denied on object object_name, database
database_name, owner owner_name."
```

If you change restricted decrypt permission from 1 to 0, the rows that reflect explicit grants remain in sysprotects. However, these rows have no effect on implicitly granted decrypt permissions because Adaptive Server does not check sysprotects to make sure decrypt permission can be implicitly granted. sp_helprotect displays misleading information for only those users who were granted or revoked explicit decrypt permission before the system was re-configured and who now have implicit decrypt permission.

Sybase recommends that you revoke any explicit decrypt permissions granted to users before you switch between enabling or disabling restricted decrypt permission to keep the system consistent.

Assigning privileges for restricted decrypt permissions

If you are using restricted decrypt permission, you can assign the privileges for creating the task's schema and managing keys as follows:

- System Security Officer – initially sets the system encryption password
- System Security Officer – configures restricted decrypt permission.
- System Security Officer – creates encryption keys and grants select permission on keys to the DBO
- DBO – creates the schema and loads data.
- System Security Officer – grants decrypt permission to the end user

When restricted decrypt permissions is set to 1, key custodians still have implicit create encryption key permissions.

New datatypes supported

Table 2-2 describes the new datatypes supported and the on-disk length of encrypted columns for datatypes supported for Adaptive Server version 15.0.2.

Table 2-2: Datatype length for encrypted columns

Datatype	Input data length	Encrypted column type	Max encrypted data length (no init_vector)	Actual encrypted data length (no init vector)	Max encrypted data length with init_vector	Actual encrypted data length (with init_vector)
date	4	varbinary	17	17	33	33
time	4	varbinary	17	17	33	33
smalldatetime	4	varbinary	17	17	33	33
datetime	8	varbinary	17	17	33	33
smallmoney	4	varbinary	17	17	33	33
money	8	varbinary	17	17	33	33
bit	8	varbinary	17	17	33	33
bigint	8	varbinary	17	1 7	33	33
unsigned bigint	8	varbinary	17	17	33	33
unichar(10)	2 (1 unichar character)	varbinary	33	17	49	33
unichar(10)	20 (10 unichar characters)	varbinary	33	33	49	49
univarchar(20)	20 (10 unichar characters)	varbinary	49	33	65	49

text, image, and unitext datatypes are not supported for this release of Adaptive Server.

Auditing enhancements for encrypted columns

Auditing for encrypted columns includes these changes:

- Auditing actions of the key custodian
- Additional audit events for alter encryption key
- Masking passwords in command text auditing

Auditing key custodian actions

Use this command to audit all actions in which keycustodian_role is active:

```
sp_audit "all", "keycustodian_role", "all", "on"
```

Additional auditing events

The encryption_key audit option writes these events and information to the audit table for encrypted columns.

Table 2-3: Auditing events for the encryption_key audit option

Command to be audited	Event number	Information in extrainfo output
create encryption key	107	Type of password, user or system
AEK as/not default	108	Modify encryption key to be the database default key or remove the default property from the key.
AEK modify encryption	118	modify encryption with user passwd or with login passwd [for user *user_name*] [with *keyvalue* [for recovery]]
AEK add encryption	119	add encryption for user user_name with user passwd [for recovery] or for login association [with keyvalue]
AEK drop encryption	120	drop encryption [for recovery] [for user]
AEK modify owner	121	modify owner new owner *user_name*
AEK key recovery	122	recover key [with keyvalue]

Masking passwords in command text auditing

Passwords are masked in the audit records produced by command text auditing. For example, if the SSO has enabled command text auditing for user “alan” in database db1:

```
sp_audit "cmdtext", "alan", "db1", "on"
```

And “alan” issues this command:

```
create encryption key key1 with passwd "bigsecret"
```

Adaptive Server writes the following SQL text to the extrainfo column of the audit table:

```
"create encryption key key1 with passwd "xxxxxx"
```

Encryption and *set proxy*

If a user issues set proxy to assume the permissions, login name, and suid of another user, Adaptive Server checks the proxy user's access to database objects, rather than the original user's access. Adaptive Server uses the name and password information of the user who logged in to check for automatic access to encryption keys using login credentials. Adaptive Server does not have access to the proxy user's password. Access to keys through the login password is on behalf of the user who logs in, not on behalf of the user assumed through an alias, set proxy, or setuser. Access to copies of encryption keys that were set up for login association, but which are still encrypted by the system encryption password, is treated similarly.

Component Integration Services and encrypted columns

Encrypted columns in Adaptive Server version 15.0.2 allows Component Integration Services (CIS) to establish connections to servers that require user-specified passwords to decrypt data. Adaptive Server forwards the user encryption key passwords from the CIS-enabled server to the remote Adaptive Server.

For example, to process a customer's credit card data, execute:

```
set encryption passwd 'MotherofSecrets'
for customer.cust_cc
```

When you select data from the table:

```
select cust_cc from customer where custid = 99
```

CIS forwards the user encryption key passwords to the remote servers.

- If CIS connections are not currently established – if the remote server is version 15.0.2 or later, when the "local" server (the server originating the CIS connections) receives a set encryption passwd statement, information contained in the statement is stored in the current thread's session context. When the local server makes a CIS connection, all context relating to encryption passwords is forwarded to the remote server.

- If CIS connections are established – CIS maintains session-state variables to be synchronized between the local and the remote server. CIS tracks which encrypted passwords are sent to the remote server. Each time it accesses the remote server, CIS compares the state information with the local thread's context, and forwards only those encryption passwords set since the last interaction with the remote server.

Changes to *ddlgen*

Adaptive Server version 15.0.2 includes changes to the ddlgen utility for encrypted columns. See Chapter 19, "Changes to ddlgen," for more information.

Changes for replication

The new encrypted columns functionality in Adaptive Server 15.0.2 is supported by replication. See the latest Replication Server documentation for more information.

Changes to commands

This section describes changes to commands for encrypted columns.

Changes to dbcc checkcatalog

In Adaptive Server 15.0.2, dbcc checkcatalog ensures that:

- The corresponding base key is present in sysencryptkeys for every key copy in sysencryptkeys. If the corresponding base key is not present, Adaptive Server issues an error.
- For every key copy, the corresponding uid is present in sysusers. If the corresponding uid is not present, Adaptive Server issues an error.

- For every decrypt default defined on a column, that the corresponding decrypt default is present in sysobjects. If the corresponding decrypt default is not present, Adaptive Server issues an error.

Changes to create encryption key

Adaptive Server 15.0.2 includes allows you to create user-specified passwords on keys with create encryption key. The syntax is:

```
create encryption key [ db.[owner ]. ] keyname  [ as default ]
[ for algorithm_name ] [ with { [ keylength num_bits ]
[ passwd 'password_phrase' ] [ init_vector {NULL | random } ]
[ pad {NULL | random } ] } ]
```

For more information, see “Key protection using user-specified passwords” on page 10.

Changes to drop encryption key

Adaptive Server version 15.0.2 drops the key copies when you drop the base key.

Changes to alter table decrypt

create and alter table include the optional decrypt_default clause, which allows you to declare a *decrypt_default* value on an encrypted column.

Changes to alter encryption key

The syntax for alter encryption key is:

```
alter encryption key [database_name.[ owner ] . ] key_name
        { [ as | not default ]
            | [ with passwd
                'password' | system_encr_passwd | login_password ]
                modify encryption
                [with passwd
                'password' | login_passwd |
                system_encr_passwd ]
             | with passwd 'password'
      add encryption [ with passwd 'password' ]
```

```
                for user user_name
                [ for login_association | for recovery ]
            | drop encryption for
                { user user_name | key recovery }
            | [ with passwd 'password ']
                recover encryption with passwd 'password'
            | modify owner user_name
}
```

- *key_name* – name for a column encryption key.
- as [not] default – indicates that the database default property should be assigned to, or unassigned from, this key.
- with passwd – specifies the current password Adaptive Server uses to decrypt the column encryption key. If you do not specify with passwd, the default is system_encr_passwd.
- *password* – specifies the new password Adaptive Server uses to encrypt the key or key copy. *password* can up to 255 bytes long.
- *login_passwd* – the login password for the user. *login_password* can be up to 255 bytes long.
- system_encr_passwd – the system encryption password for the current database.
- modify encryption – indicates you are modifying the encryption key or key copy.
- add encryption – adds encrypted key copy for a designated user.
- for user – specifies the user for whom you are adding or dropping a key copy.
- for login_association – indicates that the key copy being added is encrypted by the assigned user's login password during his or her first access to this key.
- for recovery – indicates this key copy is for recovery purposes.
- drop encryption – indicates that you are dropping the key copy for the specified user.
- recover encryption – makes the base key accessible to a new password.
- modify owner – changes the key's owner to the specified user.

Examples

Example 1

This example changes the password to the important_key encryption key. If the key owner executes this command, it reencrypts the base key; if the user assigned a key copy executes this command, it reencrypts that key copy.

```
alter encryption key important_key with passwd 'oldpassword'
modify encryption with passwd 'newpassword'
```

Example 2

Changes the password on a key copy to the current session's login password. This command can be executed only by a user who has been assigned a key copy:

```
alter encryption key important_key
     modify encryption
     with passwd login_passwd
```

Example 3

Changes the password for the important_key encryption key to the system password. This command can be executed only by the key owner or a user with sso_role, and is allowed only if a key has no key copies. It modifies the encryption of the base key:

```
alter encryption key important_key
with passwd 'ReallyBigSecret'
modify encryption with passwd system_encr_passwd
```

Example 4

Changes the password for the important_key encryption key from the system encryption password to a new password (because the system encryption password is the default password, it does not need to be specified in the statement):

```
alter encryption key important_key
     modify encryption
     with passwd 'ReallyNewPassword'
```

Example 6

Adds encryption for user "ted" for the important_key encryption key with the password `just4now`:

```
alter encryption key important_key
     with passwd 'TopSecret'
     add encryption with passwd 'just4now'
     for user 'ted'
```

Example 7

Modifies the encryption for user "ted" to use a new password. Only "ted" can execute this command:

```
alter encryption key important_key
     with passwd 'just4now'
     modify encryption
     with passwd 'TedsOwnPassword'
```

Example 8

Drops encryption for user “ted” for the `important_key` encryption key:

```
alter encryption key important_key
     drop encryption for user 'ted'
```

Example 9

Modifies the owner of important_key to new owner, “tinnap:”

```
alter encryption key important_key modify owner tinnap
```

Usage

- If the SSO issues the alter encryption key command to set the key as the database default, the specified key replaces any existing key as the default.
- If the key custodian issues alter encryption key to set a key as the database default, the specified key and the current default key (if it exists) must be owned by the key custodian.
- If you do not include the with passwd parameter with alter encryption, Adaptive Server uses the system-encryption password.
- You cannot use the system-encryption password to alter the base key of a key that has copies, and you cannot encrypt copies of keys with the system encryption password.
- Users implicitly modify only their own key-copies.
- If you specify for `login_association`, Adaptive Server temporarily encrypts the key copy with the system encryption password.
- You cannot specify for recovery and *login_association* for the same key copy.

Changes to system procedures

This section discusses the changes that have been made to system procedures for encrypted columns.

sp_encryption

Reports encryption information.

syntax

Syntax for sp_encryption:

```
sp_encryption [ help | helpkey ]
  sp_encryption [ help | helpkey ] [, key_name | wildcard ]
                [, all_dbs | key_copy | display_cols ]
  sp_encryption [ help | helpkey ] [, system_encr_password ]
                [,display_keys ]
  sp_encryption [ helpcol ] [, table_name | column_name ]
  sp_encryption [ helpuser ] [, user_name | wildcard ] [, key_copy ]
```

Parameters

- helpkey – lists encryption key properties, including:
  - Whether the database contains encryption keys.
  - The following, when run by a user with sso_role, key custodian, or DBO: keyname, keyowner, key length, key algorithm, key type, pad, initialization vector, type of password used to encrypt the key, whether key recovery has been enabled and count of key copies. The output is sorted on owner.keyname. When run by a non-privileged user, this command will list keyname, keyowner and keytype.
- help – included for backward compatibility. Includes the same output as helpkey.
- *key_name* – name of the key you are investigating. Lists the properties defined for *key_name*. If *key_name* is omitted, lists properties for all keys.
- *wildcard* – lists the properties for keys matching the wildcard pattern in the current database.
- all_dbs – lists information on encryption keys in all available databases. Only the SSO can run all_dbs.
- key_copy – lists all user copies for the specified key in the current database. The output is sorted by *key_owner*.*key_name*. Includes information about:
  - The base key owner.
  - If the key copy is a recovery key copy.
  - The user to whom a copy belongs.
  - If the copy is encrypted with a user-encryption password, a login password, or the system encryption password for login association (indicated by Login Access).

- display_cols – displays the key name, all keys (or matching wildcard keys) in the current database and the columns the key encrypts. When SSO includes display_cols, it displays columns encrypted by the keys across all available databases. When a user without the sso_role runs display_cols, only those columns encrypted by the key in the current database are displayed. Data is sorted by *key_name*, *key_owner*, *database_name*, *table_owner*, *table_name*, and *column_name*.
- helpcol *column_name*– displays the column name and the key used to encrypt the column. If the SSO includes helpcol, it prints the key name even if the key is not present in the current database. If a non-SSO user includes helpcol, Adaptive Server prints the keyid of the key if it is not present in the current database, omitting the *key_name*. The output includes: *owner.table.column*, *database.owner.keyname*. The information is sorted by *owner.table.column*.
- helpuser – displays the keys owned by or assigned to a user in the current database.
- system_encr_passwd – displays the keys and key copies that are encrypted using the system encryption password in the current database.
- display_keys – used with *system_encr_passwd* to display the keys and key copies that are encrypted using the system encryption password.

Examples

Example 1 This displays properties of all base encryption keys in the current database when run by the SSO, key custodian, or the DBO:

```
sp_encryption helpkey
Key Name    Key Owner   Key Length   Key Algorithm
Key Type              Pad         Init Vector    Type of Password
Key Recovery     # of Key Copies
----------  --------    ---------    --------------
----------           -------     ------------   -----------
--------         ------
tinnap_key  tinnap       128          AES
Symmetric key         0           1              System Encr Passwd
0             0
tinnap_key1 tinnap       128          AES
Symmetric default key 0            1             User Passwd
1             3
sample_key1  dbo         192          AES
Symmetric key 1        1                         Login Passwd
1               2
```

When run by user “tinnap,” this displays the following properties of all base encryption keys in the current database:

```
sp_encryption helpkey
Key Name          Key Owner   Key Type
--------------     ---------   ------------
tinnap_key         tinnap      Symmetric key
tinnap_key1        tinnap      Symmetric default key
sample_key1         dbo        Symmetric key
```

Example 2

Displays properties of all base encryption keys with names similar to “tinnap%” in the current database when run by SSO, key custodian, or DBO:

```
sp_encryption helpkey, "tinnap%"
Key Name   Key Owner   Key Length   Key Algorithm   Key Type
Pad     Init Vector    Type of Password       Key Recovery
# of Key Copies
---------  --------    ---------     -------------- ---------
-----   -----------   --------------          -----------
--------------
tinnap_key   tinnap      128            AES          Symmetric key
0       1             System Encr Passwd       0
0
tinnap_key1  tinnap      128            AES           Symmetric default key
0       1             User Passwd              1              3
```

When run by user “tinnap,” displays the following properties for the base encryption keys in the current database with names similar to “tinnap%”:

```
sp_encryption helpkey, "tinnap%"
Key Name           Key Owner  Key Type
------------------  ---------  ------------
tinnap_key          tinnap     Symmetric key
tinnap_key1         tinnap     Symmetric default key
```

Example 3

Displays the properties of base encryption key sample_key1 when run by the SSO, key custodian, or DBO in the current database:

```
sp_encryption helpkey, sample_key1
Key Name     Key Owner   Key Length  Key Algorithm  Key Type
Pad          Init Vector   Type of Password   Key Recovery
# of Key Copies
-----------  --------    ----------  -------------  ---------  -----
-----------  -----------  ------------      ---------
```

```
--------------
sample_key1  dbo          192          AES             Symmetric Key
1             1            Login              1
2
```

When non-privileged user "tinnap" runs this command, it displays the following properties for the base encryption key sample_key1 in the current database:

```
sp_encryption helpkey, sample_key1
Key Name       Key Owner    Key Type
-------------  -----------  ------------
sample_key1    dbo          Symmetric key
```

Example 4

Displays the properties of all base encryption keys in all available databases (only the SSO can run this command):

```
sp_encryption helpkey, NULL, all_dbs
Db.Owner.Keyname            Key Length  Key Algorithm  Key Type
Pad    Init Vector   Type of Password     Key Recovery    #of Key Copies
-----------------------   ----------   -----------  --------------------
-----  ----------   ---------------       ------          --------------
keydb.dbo.cc_key            256           AES       Symmetric default key
1      1            System EncrPasswd     0              0
keydb.dbo.sample_key1       128          AES        Symmetric key
0      0            System Encr Password  1              4
keydb1.tinnap.tinnap_key    128          AES        Symmetric key
0      1            System Encr Passwd    0              0
keydb1.tinnap.tinnap_key1   128          AES        Symmetric default key
0      1            User Password         1              3
keydb1.dbo.sample_key1      192          AES        Symmetric key
1      1             Login Passwd         1              2
```

Example 5

Displays the properties of all base encryption keys similar to %key1 in all available databases (only the SSO can run this command):

```
sp_encryption helpkey, '%key', all_dbs
Db.Owner.Keyname            Key Length  Key Algorithm  Key Type
Pad    Init Vector   Type of Password     Key Recovery    #of Key Copies
-----------------------   ----------   -----------  --------------------
-----  ----------   ---------------       ------          -----
keydb.dbo.cc_key            256           AES       Symmetric default key
1      1            System EncrPasswd     0              0
keydb1.tinnap.tinnap_key    128          AES        Symmetric key
0      1            System Encr Passwd    0              0
```

Example 6

Displays the properties of base encryption key sample_key1 in all available databases (only the SSO can run this command):

```
sp_encryption helpkey, sample_key1, all_dbs
Db.Owner.Keyname           Key Length  Key Algorithm  Key Type
Pad    Init Vector   Type of Password     Key Recovery   #of Key Copies
-----------------------   ----------   -----------  -------------------
-----  ----------   ---------------       ------         -----
keydb.dbo.sample_key1      128          AES        Symmetric key
0      0            System Encr Password  1             4
keydb1.dbo.sample_key1     192          AES        Symmetric key
1      1             Login Passwd          1             2
```

Example 7

Displays all the user access copies of keys when run by the SSO, key custodian, or DBO in the current database:

```
sp_encryption helpkey, Null, key copy
Owner.Keyname         Assignee    Type of Password          Key Recovery
------------------    ----------   ---------------           ----------
tinnap.tinnap.key1    joesmp       User Passwd               0
tinnap.tinnap.key1    samcool      User Passwd               1
tinnap.tinnap.key1    billyg       User Passwd               0
dbo.sample.key1       tinnap       Login Access              0
dbo.sample.key1        joesmp       Login Passwd             1
```

When user “tinnap” runs this command, it displays the key copies assigned to this user and the key copies for the keys “tinnap” owns in the current database:

```
sp_encryption helpkey, Null, key copy
Owner.Keyname         Assignee    Type of Password          Key Recovery
------------------    ----------   ---------------           ----------
tinnap.tinnap.key1    joesmp       User Passwd               0
tinnap.tinnap.key1    samcool      User Passwd               1
tinnap.tinnap.key1    billyg       User Passwd               0
dbo.sample.key1       tinnap       Login Access              0
```

Example 8

Displays all the user access copies of keys with name similar to “sample%” when run by the SSO, key custodian, or DBO:

```
sp_encryption helpkey, "sample%", key copy
Owner.Keyname        Assignee         Type of Password       Key Recovery
------------------    ----------       ---------------         ----------
dbo.sample_key1        tinnap            Login Access            0
dbo.sample_key1        joesmp             Login Passwd           1
```

When user “tinnap” runs this command, it displays the key copies of keys with names similar to “sample%” assigned to user “tinnap,” and the key copies for keys with names similar to “sample%” for which “tinnap” is the owner in the current database:

```
sp_encryption helpkey, "sample%", key copy
Owner.Keyname         Assignee          Type of Password       Key Recovery
------------------    ----------        ----------------       ----------
dbo.sample_key1        tinnap            Login Access          0
```

Example 9

When run by the SSO, key custodian, or the DBO, displays all key copies for key tinnap_key1 in the current database:

```
sp_encryption helpkey, tinnap_key1, key copy
Owner.Keyname          Assignee         Type of Password       Key Recovery
------------------      ---------        -----------------      ----------
tinnap.tinnap_key1      joesmp           User Passwd            0
tinnap.tinnap_key1      samcool          User Passwd            1
tinnap.tinnap_key1       billyg           User Passwd           0
```

When run by user “joesmp,” this displays all encryption key copies assigned to user “joesmp” and also all the key copies for that keyname if the user is the owner of the key in the current database:

```
sp_encryption helpkey, tinnap_key1, key_copy
Owner.Keyname          Assignee         Type of Password       Key Recovery
------------------      ---------        -----------------      ----------
tinnap.tinnap_key1      joesmp           User Passwd            0
```

Example 10

When run by the SSO, displays all encrypted columns in all available databases encrypted by keys in the current database:

```
sp_encryption helpkey, null, display_cols
Key Name        Key Owner   Database Name    Table Owner       Table Name
Column Name
------------    -------       -----------       ----------        ------------
--------------
tinnap_key      tinnap        testdb1           tinnap            t3
c3
tinnap_key1     tinnap        testdb            tinnap            t4
c4
sample_key1     dbo           coldb             dbo               t1
c1
sample_key1     dbo           coldb             billyg            t2
```

```
c2
```

When this statement is run by user "tinnap," Adaptive Server displays the columns in the current database encrypted by keys in the current database:

```
sp_encryption helpkey, null, display_cols
Key Name       Key Owner   Database Name    Table Owner      Table Name
Column Name
------------   -------      -----------      ----------       ------------
--------------
tinnap_key     tinnap       testdb1          tinnap           t3
c3
```

Example 11

When run by the SSO, displays all encrypted columns in all available databases encrypted by a key with a name like "%key%" in the current database:

```
sp_encryption helpkey, "%key%", display_cols
Key Name       Key Owner   Database Name    Table Owner      Table Name
Column Name
------------   -------      -----------      ----------       ------------
--------------
tinnap_key1    tinnap       testdb           tinnap           t4
c4
sample_key1    dbo          coldb            dbo              t1
c1
sample_key1    dbo          coldb            billyg           t2
c2
```

When this statement is run by user "tinnap," Adaptive Server returns all columns that are encrypted by keys with name matching "%key%" in the current database:

```
sp_encryption helpkey, "%key%", display_cols
Key Name       Key Owner   Database Name    Table Owner      Table Name
Column Name
------------   -------      -----------      ----------       ------------
--------------
tinnap_key1    tinnap       testdb           tinnap           t4
c4
```

Example 12

This example displays all columns which have been encrypted by key sample_key1 across all available databases:

```
sp_encryption helpkey, sample_key1, display_cols
Key Name          Key Owner         Database Name         Table Owner
```

```
Table Name   Column Name
------------       ----------       ---------------       ----------
-----------  -----------
sample_key1        dbo               coldb                dbo
t1             c1

sample_key1        dbo               coldb                billyg
t2           c2
```

When run by user "tinnap," displays all columns in the current database that are encrypted by key sample_key1:

```
sp_encryption helpkey, sample_key1, display_cols
Key Name           Key Owner         Database Name         Table Owner
Table Name   Column Name
------------       ----------       ---------------       ----------
-----------  -----------
sample_key1        dbo               coldb                dbo
t1             c1

sample_key1        dbo               coldb                billyg
t2           c2
```

Example 13

When run by the SSO, key custodian, or DBO, lists keys and key copies that are encrypted with the system encryption password in the current database:

```
sp_encryption helpkey, system_encr_passwd, display_keys
Owner.Keyname                    Assignee
---------------                   -------------
dbo.cc_key                        NULL
dbo.sample_key1                   NULL
dbo.sample_key1                   tinnap
```

When run by user "tinnap," this command displays the keys owned by, or key copies assigned to, user "tinnap" that are encrypted with the system encryption password in the current database:

```
sp_encryption helpkey, system_encr_passwd, display_keys
Owner.Keyname                    Assignee
---------------                   -------------
dbo.sample_key1                  tinnap
```

Example 14

Lists all base keys owned by users in the current database when the SSO, key custodian, or DBO runs this command:

```
sp_encryption helpuser
```

```
Owner.Keyname                Type of Password
---------------              -------------------
tinnap.tinnap_key            System Encr Passwd
tinnap.tinnap_key1           User Passwd
dbo.sample_key1              Login Passwd
```

If user "tinnap" runs this command, lists all base keys owned by this user in the current database:

```
sp_encryption helpuser
Owner.Keyname                Type of Password
---------------              -------------------
tinnap.tinnap_key            System Encr Passwd
tinnap.tinnap_key1           User Passw
d
```

Example 15

When run by the SSO, key custodian, or DBO, lists all base encryption keys owned by user "tinnap" in the current database:

```
sp_encryption helpuser, tinnap
Owner.Keyname                Type of Password
---------------              -------------------
tinnap.tinnap_key            System Encr Passwd
tinnap.tinnap_key1           User Passwd
```

If run by user "tinnap," lists all base encryption keys owned by user "tinnap" in the current database:

```
sp_encryption helpuser, tinnap
Owner.Keyname                Type of Password
---------------              -------------------
tinnap.tinnap_key            System Encr Passwd
tinnap.tinnap_key1           User Passwd
```

Example 16

When run by the SSO, key custodian, or DBO, lists all key copies assigned to all users in the current database:

```
sp_encryption helpuser, NULL, key_copy
Owner.Keyname            Assignee        Type of Password     Key Recovery
---------------------     ----------      ----------------   ---------
dbo.sample_key1            tinnap          Login Passwd          0
tinnap.tinnap_key1         joesmp          User Passwd           0
dbo.sample_key1            joesmp          Login Passwd          1
tinnap.tinnap_key1         samcool         User Passwd           1
tinnap.tinnap_key1          billyg          User Passwd          0
```

If user “tinnap” runs this statement, it displays the key copies assigned to this user and the key copies for the keys owned by this user in the current database:

```
sp_encryption helpuser, NULL, key_copy
Owner.Keyname            Assignee        Type of Password    Key Recovery
---------------------     -----------      ----------------   ---------
dbo.sample_key1           tinnap           Login Passwd       0
tinnap.tinnap_key1        joesmp           User Passwd        0
tinnap.tinnap_key1        samcool          User Passwd        1
tinnap.tinnap_key1        billyg           User Passwd        0
```

Example 17

When run by the SSO, key custodian, or DBO, lists all the key copies in the current database with assignee names like “%na%”:

```
sp_encryption helpuser, '%na%', key_copy
Owner.Keyname          Assignee         Type of Password      Key Recovery
---------------------   -----------      ------------------    -----------
dbo.sample_key1         tinnap           Login Passwd          0
tinnap.tinnap_key1      joesmp           User Passwd           0
dbo.sample_key1         joesmp           Login Passwd          1
```

When run by user “tinnap,” lists all the key copies in the current database with assignee name like “%na%” and the key copies for keys owned by this user with name like “%na%” only if the user’s name matches the wildcard pattern:

```
sp_encryption helpuser, '%na%', key_copy
Owner.Keyname          Assignee         Type of Password      Key Recovery
---------------------   -----------      ------------------    -----------
dbo.sample_key1         tinnap           Login Passwd          0
tinnap.tinnap_key1      joesmp           User Passwd           0
tinnap.tinnap_key1      samcool          User Passwd           1
tinnap.tinnap_key1      billyg           User Passwd           1
```

Example 18

When run by the SSO, key custodian, or DBO, lists all encrypted columns in the current database (coldb in this example) and the keys used to encrypt the columns:

```
sp_encryption helpcol
Owner.Table.Column           Db.Owner.Keyname
-----------------------       ---------------------
dbo.t1.c1                     keydb1.dbo.sample_key1
billyg.t2.c2                  keydb.dbo.sample_key1
tinnap.t3.c3                  coldb.dbo.sample_key2
```

When user tinnap runs this statement in the coldb database, Adaptive Server displays values for keyid instead of keyname for those keys not in coldb:

```
sp_encryption helpcol
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                     keydb1.123456
billyg.t2.c2                  keydb.2345678
tinnap.t3.c3                  coldb.dbo.sample_key3
```

Example 19

When run by the SSO, lists all encrypted columns in table t3 in the current database, and the keys used to encrypt the columns across all available databases:

```
sp_encryption helpcol, t3
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
tinnap.t3.c3                 coldb.dbo.sample_key2
```

When run by user "tinnap," lists all encrypted columns in table t3 in the current database and the keys used to encrypt the columns:

```
sp_encryption helpcol, t3
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
tinnap.t3.c3                 coldb.dbo.sample_key3
```

Example 20

When run by the SSO, lists all encrypted columns named c1 in the current database across all available databases, and the keys used to encrypt the columns:

```
sp_encryption helpcol, c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.dbo.sample_key1
```

When run by user "tinnap," lists all encrypted columns named c1 in the current database and the keyid of the keys used to encrypt the columns if the key is not present in the current database:

```
sp_encryption helpcol, c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.123456
```

Example 21

When run by the SSO, lists all encrypted columns in table dbo.t1 in the current database and the keys used to encrypt the columns across all available databases:

```
sp_encryption helpcol, dbo.t1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.dbo.sample_key1
```

When run by user "tinnap," lists all encrypted columns in table dbo.t1 in the current database and the keyid of the keys used to encrypt the columns if the key is not present in the current database:

```
sp_encryption helpcol, dbo.t1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.123456
```

Example 22

When run by the SSO, lists all encrypted columns named c1 in table t1 in the current database and the keys used to encrypt the columns across all available databases:

```
sp_encryption helpcol, t1.c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.dbo.sample_key1
```

When run by user "tinnap," lists all encrypted columns named c1 in table t1 in the current database and the keyid of the keys used to encrypt the columns if the key is not present in the current database:

```
sp_encryption helpcol, t1.c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.12345678
```

Example 23

When run by the SSO, lists all encrypted columns named c1 in table t1 owned by the DBO in the current database, and the keys used to encrypt the columns across all available databases:

```
sp_encryption helpcol, dbo.t1.c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.dbo.sample_key1
```

When run by user "tinnap," lists all encrypted columns named c1 in table t1 owned by the DBO, and the keyid of keys used to encrypt the columns if the key is not present in the current database:

```
sp_encryption helpcol, dbo.t1.c1
Owner.Table.Column           Db.Owner.Keyname
-----------------------      --------------------
dbo.t1.c1                    keydb1.123456789
```

Usage

1 Which user runs sp_encryption determines its output. If a user without privileges runs sp_encryption helpkey, it reports the *key_name*, *key_owner*, and *key_type* for all base keys in the current database. When the SSO, key custodian, or DBO run sp_encryption helpkey, it lists these properties for all base keys in the current database: key name, key owner, key length, key algorithm, key type, pad, initialization vector, type of password used to encrypt the key, whether key recovery is enabled, and the number of key copies. The output is sorted on *key_name* and *key_owner*. *key_type* indicates if the key is symmetric and has the default property. The type of password can be a system encryption password or a user encryption password.

2 If you run sp_encryption helpkey and no keys are present in the database, it returns an informational message saying there are no keys in the current database.

3 When the SSO, key custodian, or DBO runs sp_encryption helpkey, *key_name*, it lists the base keys in the current database for *key_name*. If it is run by a user without privileges, it lists the *key_owner*, *key_name* and *key_type* for this *key_name*.

4 When the SSO runs sp_encryption helpkey, *key_name*, *display_cols*, it lists all columns across all available databases encrypted by *key_name*. If it is run by a user without privileges, it lists the columns in the current database encrypted by *key_name*.

If the SSO runs sp_encryption helpkey, *key_name*, *display_cols* and the *key_name* value is NULL, it displays all encrypted columns across all available databases. When run by a user without privileges, it displays all encrypted columns in the current database.

5 Only an SSO can run sp_encryption helpkey, [,*key_name* | wildcard], all_dbs to get the properties of keys in all databases. If a user without the sso_role runs this command, they receive an "unauthorized user" error message. If no keys qualify the keyname or wildcard, Adaptive Server returns a message stating `'There are no encryption keys (key copies) like keyname in all databases'.`

6 You must specify the *key_copy* parameter to get information about key copies. If you do not specify the *key_copy* parameter, sp_encryption returns information only about base keys.

7 If *key_name* is NULL in sp_encryption helpkey, key_name, key_copy, it lists all the key copies in the current database for a SSO, key custodian, or DBO. If it is run by a user without privileges, it lists all the key copies assigned to the user in the current database and all key copies of the keys owned by the user in the current database.

8 When a SSO, key custodian, or DBO runs sp_encryption helpkey, *key_name*, *key_copy*, it lists the key copies in the current database for *key_name*. If this is run by a user without privileges, it lists the key copies assigned to the user for that *key_name* and the key copies for that *key_name* if the user is the key owner.

9 The SSO, key custodian, and DBO can run sp_encryption helpkey, *system_encr_passwd*, display_keys to receive information on all keys and key copies in the current database encrypted by system encryption password. Users without privileges receive information about the base encryption keys or key copies they own or are assigned in the current database. Key copies are encrypted with the system encryption password only when they are created for login association. The output is sorted by *owner.keyname*.

10 If an SSO, key custodian, or DBO runs sp_encryption helpuser, *user_name*, key_copy without specifying a *user_name* and *key_copy* for the helpuser parameter, it lists all the base keys owned by all users in the current database. If sp_encryption is run by a user without privileges without specifying a *user_name* or *key_copy*, it displays the base keys owned by the current user.

If any user runs sp_encryption helpuser, *user_name*, it lists all the base keys owned by owner.keyname. If a user without privileges runs the command and owns no base keys, Adaptive Server displays an informational message stating this.

If an SSO, key custodian, or DBO runs sp_encryption helpuser, *user_name*, *key_copy*, it lists the key copies assigned to *user_name*. If a user without privileges issues this command, its lists the key copies assigned to this user and all the key copies of the keys owned by the user in the current database, with these columns in the result set: Owner.Keyname, Assignee, Type of Password, and Key Recovery. The output is sorted by Assignee.

If *user_name* is NULL for sp_encryption helpuser *user_name*, *key_copy*, it lists all the key copies in the current database for a SSO, key custodian, or DBO. For users without privileges, it lists all the key copies assigned to the user in the current database and the key copies for the keys owned by this user.

11 For sp_encryption helpcol, column_name, *column_name* uses the form *name.name.name*, where:

- *name* – if sp_encryption finds no tables of this name, it looks for all columns of that name
- *name.name* – is *owner.table*. If sp_encryption finds no tables of this name, it looks for a single column named *table.column*
- *name.name.name* – is *owner.table.name*

For all columns identified by these rules in the current database, sp_encryption displays column name along with the key used to encrypt the column.

The output for sp_encryption helpcol, *column_name* is *Owner.Table.Column* and *Db.Owner.Keyname*. The *key_name* is expressed as *database.keyid* when run by non-SSO users, and the key is present in a different database from the encrypted column. The result set is sorted by *Owner.Table.Column*.

sp_password

When you use sp_password to change your password, Adaptive Server:

- Decrypts all your login password encrypted key copies using the old password and then reencrypts them with the new password.
- Updates any key copies assigned to you that are designated for login association. The key copies' password type remains the same.

After a password change, you should log out of all your Adaptive Server sessions before accessing any encrypted data. Alternatively, you can use the immediate parameter of sp_password to propagate the password change to all sessions.

When the SSO issues sp_password to reset a user's password, the user's login password-encrypted key copies are dropped by Adaptive Server because the user's old login password is not available. Adaptive Server requires the key custodian to recreate the key copies for the user, if they are needed

sp_helprotect

sp_helprotect includes the *permission_name* parameter to:

- Display the grantor name.
- Display the table or column name
- Show whether the permission is grantable for all permissions granted in a given database.
- Check for permissions on commands like update, delete, select.

The value for *permission_name* can be any of the values from sysprotects.action.

You can run sp_helprotect *permission_name* on a table, view, stored procedure, SQLJ stored procedure, SQLJ function, or using name of a user, user-defined role, group, or permission name in the current database. For example, this runs sp_helprotect using the action "Decrypt" from sysprotects.action:

```
sp_helprotect "Decrypt"

grantor   grantee   type   action   object     column   grantable
-------   --------  -----  -------  -------    ------   ---------
sa1        hr_login   Grant  Decrypt  employee   ssn       TRUE
sa1        hr_role    Grant  Decrypt  employee   ssn       FALSE
```

Any user can run sp_helprotect *permission_name* to view his or her permission information. Only the SSO can view permissions for all users.

sp_dropuser

sp_dropuser drops all key copies from sysencryptkeys for the specified user in the current database.

sp_help

The result set for sp_help *table_name* includes the `Decrypt_Default_name` column, which indicates the decrypt default name for the column. For example, if you run the following:

```
create table encr_table(col1 int encrypt decrypt_default 1)
```

When you run sp_help on encr_table, it shows the following:

```
Column_name   Type     Length   Prec   Scale   Nulls   Default_name
Rule_name    Access_Rule_name Computed_Column_object  Identity     Encrypted
Decrypt_Default_name
-----------    ----    ------    ----   -----   -----   ------------
---------    --------------   --------------------    ---------  -------------
------------
c1             int     4          NULL   NULL    0        NULL
NULL         NULL             NULL                    0          1
encr_table_col1_1036527695
```

sp_helpconfig

For Adaptive Server version 15.0.2, sp_helpconfig 'restricted decrypt permission displays:

```
sp_helpconfig 'restricted decrypt permission'
0 - restricted decrypt permission disabled (default).
1 - restricted decrypt permission enabled

Minimum Value                Maximum Value   Default Value   Current Value
Memory Used      Unit       Type

 ----------------            -----------     ----------      ---------
-------    ------------      -------------
0                            1                0              0
0          switch             dynamic
```

Reserved words

Adaptive Server version 15.0.2 adds this reserved word for encrypted columns:

- encrypt

Downgrading Adaptive Server 15.0.2 to 15.0 or 15.0.1

You can downgrade encrypted columns from version 15.0.2 to version 15.0 or 15.0.1. Adaptive Server versions 15.0 GA and 15.0 ESD #1 did not support encrypted columns, so downgrading to either of these versions requires that you remove all encryption from your databases. Versions 15.0 ESD #2 and later and 15.0.1 supported encrypted columns, and downgrade to these versions requires just that you remove the 15.0.2 encrypted columns functionality from your databases.

Note If Adaptive Server was never configured for encrypted columns, you do not need to perform any steps described in this section as preparation to downgrade Adaptive Server version 15.0.2 to to15.0 or 15.0 ESD#1.

Replication issues

If you are downgrading a server in which you have enabled replication on databases that contain encrypted data, you should perform the following steps before you start the downgrade procedure:

1 Stop all activity in the databases.

2 Verify that you have successfully transferred all replicated data in the primary database transaction log to the standby or replicate database. The process for doing this depends on the application you are using. See the documentation for your application for more information.

3 Truncate the transaction log in the primary database:

```
sp_stop_rep_agent primary_dbname
dbcc settrunc ('ltm', 'ignore')
dump tran  primary_dbname with truncate_only
```

4 Set the Replication Server locator to zero. Shut down replication server. In the RSSD for the Replication Server, run:

```
rs_zeroltm primary_servername, primary_dbname
```

Downgrading from Adaptive Server 15.0.2 to 15.0 GA or 15.0 ESD #1

If you have configured Adaptive Server to use encrypted columns, you must drop or decrypt the data in the encrypted columns before you reload it on Adaptive Server version 15.0 or Adaptive Server release 15.0 ESD #1. If you do not, processing encrypted data in the earlier version of Adaptive Server results in errors or data corruption.

Note You cannot load any dumps of Adaptive Server 15.0.2 databases or transactions with encrypted columns into an earlier version of Adaptive Server 15.0 until you have performed the following steps.

To decrypt the data on all affected tables and prepare the server to be downgraded:

1 Start Adaptive Server in single-user mode to ensure that no other user can process encrypted columns while you remove encryption from the database.

2 Verify that sp_configure "enable encrypted columns" shows a run value of 1.

3 If you have configured restricted decrypt permission, turn it off:

```
sp_configure "restricted decrypt permission", 0
```

4 Generate a list of all encryption keys in all databases. You must have the sso_role to run this command:

```
sp_encryption helpkey, NULL, all_dbs
```

5 Generate a list of all columns encrypted by each key. You must have the sso_role to run this command, and you must run the command from the database where the key was created:

```
sp_encryption helpkey, key_name, display_cols
```

The result set includes the key name, key owner, database name, table owner, table name, and column name for each column encrypted by *key_name*.

Alternatively, you can run sp_help *table_name* on each table in each database. The column description indicates which columns are encrypted.

6 Either drop the tables that have encrypted columns, or run alter table to decrypt the data. You must be the table owner or have sa_role to drop the table. You must be the table owner or have the sso_role to decrypt the data.

For example, to decrypt the column cc_no in the customer table, enter:

```
alter table customer modify cc_no decrypt
```

7 Drop all encryption keys to ensure that you have removed all encryption from the database. Adaptive Server returns an error message if you attempt to drop a key that is associated with an existing encrypted column.

For example, to drop an encryption key user_key1, as key owner or as a user with the sso_role, enter:

```
drop encryption key user_key1
```

8 To determine which privileges have been granted to users with the keycustodian_role, run the following command in each database:

```
sp_helprotect keycustodian_role
```

Remove the privileges listed:

```
revoke <permission> from keycustodian_role
```

9 In each database, for each user, run:

```
sp_displayroles <username>
```

Revoke keycustodian_role from any users to whom it was granted:

```
revoke role keycustodian_role from <user1>, <user2>,
...
```

10 Drop the system encryption password in every database where it is set. Enter:

```
sp_encryption system_encr_passwd, "old password",
null
```

11 Drop sp_encryption and the auxiliary stored procedures (sp_aux_encrkey_lookup, sp_aux_encrkey_info, and sp_aux_encr_lookup).

12 Disable encrypted columns:

```
sp_configure "enable encrypted columns", 0
```

13 Execute sp_downgrade, which verifies that you have cleared your databases of all encrypted columns and prepares the server for downgrade. See “Running sp_downgrade on encrypted columns” on page 69 for more information.

Downgrading from 15.0.2 Adaptive Server to 15.0 ESD#2 or later

To downgrade Adaptive Server release 15.0.2 using encrypted columns to version 15.0 ESD #2 or later:

1 Start Adaptive Server in single-user mode to ensure that no other user can process encrypted columns while you remove encryption from the database.

2 Drop all key copies. Use sp_encryption to list the key copy assignee, the key name, owner, type of password, and whether the key copy is for recovery:

```
sp_encryption helpuser, null, key_copy
```

For example, to drop the key copies for key k1:

```
alter encryption key k1 drop encryption
for user 'joe'
alter encryption key k1 drop encryption
for user 'tinnap'
```

To drop the recovery key copy, execute:

```
alter encryption key k1 drop encryption for recovery
```

3 Modify any keys encrypted by explicit passwords to reencrypt them with the system encryption password. For example, if key k1 was created or altered for encryption using the password “rur2d2”, enter:

```
alter encryption key k1 with passwd "rur2d2"
modify encryption with passwd system_encr_passwd
```

4 Remove decrypt default attribute from encrypted columns. Use sp_help to determine whether a column has a decrypt default (for example, `sp_help employee`).

The output of sp_help indicates the internal names of any decrypt defaults and the column to which it applies.

For example, remove the decrypt_default from employee.salary, using:

```
alter table employee replace salary drop
```

```
decrypt_default
```

5 Decrypt any columns whose datatype is not supported for encryption in versions earlier than 15.0.2. These include columns of type datetime, money, bigint, bit, unichar, univarchar, smalldatetime, smallmoney, date, time, and unsigned bigint. The output of sp_help indicates the types of the columns that are marked as encrypted.

   For example, decrypt the salary column as:

   ```
   alter table employee modify salary decrypt
   ```

6 Run the following command in each database:

   ```
   sp_helprotect keycustodian_role
   ```

   Revoke any privileges granted to keycustodian role:

   ```
   revoke <permission> from keycustodian_role
   ```

7 For every user, run this in every database:

   ```
   sp_displayroles <username>
   ```

   Revoke keycustodian_role from any users to whom it was granted:

   ```
   revoke role keycustodian_role from <user1>, <user2>,
   ...
   ```

   Running this command does not affect ownership of keys. Any users who were not previously explicitly granted create encryption key permissions cannot create new keys unless the SSO grants create encryption key to them.

8 If you have increased the size of the system encryption password to more than 64 bytes in version 15.0.2, you must modify it back to a pre-15.0.2 length. Enter:

```
sp_encryption system_encr_passwd 'old...long...password....',
'new...shorter...password'
```

9 Disable restricted decrypt permission:

   ```
   sp_configure 'restricted decrypt permission', 0
   ```

10 Execute sp_downgrade, which verifies that you have cleared your databases of all encrypted columns and prepares the server for downgrade. See Running sp_downgrade on encrypted columns, below, for more information.

Running sp_downgrade on encrypted columns

sp_downgrade prepares your system for downgrade to a previous 15.0.x version. See Chapter 6, "Downgrading from Adaptive Server 15.0.2" in the *Installation Guide* for more information.

The examples in this section illustrate how sp_downgrade output applies to encrypted functionality used in Adaptive Server version 15.0.2. It associates the sp_downgrade output with the necessary downgrade actions, which are listed in “Downgrading from Adaptive Server 15.0.2 to 15.0 GA or 15.0 ESD #1” on page 65 and “Downgrading from 15.0.2 Adaptive Server to 15.0 ESD#2 or later” on page 67.

You must run sp_downgrade twice: the first time to check if your system is ready for downgrade and the second time to make changes so that your system can be used with Adaptive Servers earlier than 15.0.2. The syntax is:

```
sp_downgrade 'prepare' | 'downgrade' | 'help' , 'version_number'
```

where *version_number* must be either `15.0` or `15.0.1`. If you are downgrading to:

- 15.0 GA or 15.0 ESD #1, enter “15.0”. For example:

  ```
  sp_downgrade 'prepare', '15.0'
  ```

- 15.0 ESD #2 or later, enter “15.0.1”. For example:

  ```
  sp_downgrade 'prepare', '15.0.1'
  ```

Preparing to downgrade

If you have not already prepared your server for downgrade by following the steps in either “Downgrading from Adaptive Server 15.0.2 to 15.0 GA or 15.0 ESD #1” on page 65 or “Downgrading from 15.0.2 Adaptive Server to 15.0 ESD#2 or later” on page 67, running sp_downgrade 'prepare' may produce one or more of several messages. For example, running:

```
sp_downgrade 'prepare', '15.0'
```

Return this output:

```
Encrypted Columns Item          Required  Action
------------------------------  ------------------------
master                          Revoke keycustodian_role from user
testdb.dbo.v4.col2              Alter table to drop decrypt default
testdb.dbo.v4.col2              Alter table to decrypt column
```

Table 2-4 shows the encrypted columns functionality that causes the error message, along with the action you must take in one or more databases to prepare your system for downgrade.

Table 2-4: sp_downgrade error messages

Downgrading to version:	Encrypted columns conditions being checked	Error message output	Action to perform
15.0.1	Existence of key copies in any database	`"Alter key to drop key copy"`	Run alter encryption key...drop encryption ... for user to drop key copies of the named keys.
15.0.1	Existence of recovery key copy in any database	`"Alter key to drop key recovery"`	Run alter encryption key...drop encryption ... for recovery to drop recovery copies of the named keys.
15.0.1	Keys encrypted by user-specified password	`"Alter key to encrypt by system encryption password"`	Use alter encryption key...modify encryption to change the named keys for encryption by system encryption password.
15.0 or 15.0.1	Existence of permissions granted to keycustodian_role	`"Revoke privileges from role"`	Use revoke command to revoke privileges granted to keycustodian_role in the named database.
15.0 or 15.0.1	keycustodian_role granted to user	`"Revoke keycustodian_role from user"`	Use revoke command to revoke keycustodian_role from users in named database.
15.0 or 15.0.1	System encryption password longer than 64 bytes	`"Reset system encryption password with length <= 64 bytes"`	Use sp_encryption to change the system encryption password to one that less than or equal to 64 bytes long.
15.0 or 15.0.1	Existence of decrypt_default property on a column	`"Alter table to drop decrypt default"`	Use alter table ... replace to remove the decrypt default property form the named column.
15.0 or 15.0.1	Setting of restricted decrypt permission configuration parameter.	`"Turn off the configuration parameter resticted decrypt permission"`	Use sp_configure to set restricted decrypt permission to 0
15.0 or 15.0.1	Presence of encrypted columns	`"Alter table to decrypt column"`	Use alter table .. modify to decrypt named column. For 15.0.2 you are asked to decrypt only those columns whose type is not supported for encryption in 15.0.1.

You can run sp_downgrade 'prepare' as many times as is necessary to validate that you have reduced the encrypted columns functionality to the required level. When sp_downgrade detects that all column encryption items have been addressed for the specified downgrade version, it prints the message `"There are no errors, databases are ready for downgrade."`

Executing sp_downgrade

The downgrade phase of sp_downgrade takes various actions in the server to allow the system to be started with a pre-15.0.2 server. There are no specific actions you need to take. To start the downgrade phase, run:

```
sp_downgrade 'downgrade', '15.0.1'
```

Or:

```
sp_downgrade 'downgrade', '15.0'
```

CHAPTER 3

New Case-Insensitive Sort Order For Chinese and Japanese Character Sets

This chapter contains information about case-sensitive sort orders supported in Adaptive Server version 15.0.2.

Overview

This chapter describes case-insensitive sort orders for these Chinese and Japanese character sets:

- EUC-GB
- GB-18030
- CP-936
- EUC-JIS
- SJIS
- DECKANJI

New available sort orders

Table 3-1 shows the sort orders available for Simplified Chinese and Japanese.

Table 3-1: Available sort orders

Language or script	Character sets	Sort orders
Simplified Chinese	EUC-GB, GB-18030, CP936	General purpose case-insensitive dictionary ordering
Japanese	EUCJIS, SJIS, DECKANJI	General purpose case-insensitive dictionary ordering

Selecting case-insensitive sort orders for Chinese and Japanese character sets

Use two stored procedures to select case-insensitive sort orders:

- sp_helpsort
- sp_configure

sp_helpsort

sp_helpsort lists the available case-insensitive sort orders.

```
sp_helpsort
-----------
Name                    ID
-------------------------
nocase_eucgb            52
nocase_cp936            52
nocase_gb18030          52
nocase_eucjis           52
nocase_sjis             52
nocase_deckanji         52
```

sp_configure

To switch to a case-insensitive sort order, enter:

```
sp_configure 'default sortorder id', 52
```

CHAPTER 4

Archive Database Access

Archive database access allows a database administrator to validate or selectively recover data from a database dump (an “archive”) by making the dump appear as if it were a traditional read-only database; this type of database is called an “archive database.”

Unlike a traditional database, an archive database uses the actual database dump as its main disk storage device, with a minimum amount of traditional storage to represent new or modified pages that result from the recovery of the database dump. A database dump already contains the images of many (if not most) of the database pages, therefore, an archive database can be loaded without having to use Backup Server to transfer pages from the archive to traditional database storage. Consequently, the load is significantly faster than a traditional database.

Overview

Archive database access enables a variety of operations to be performed directly on a database dump.

The amount of storage needed for a traditional database load must be equal to or greater than the size of the source database; the loading of the database dump using Backup Server involves copying pages from the database dump into the storage that has been set aside for the traditional database.

By contrast, you can create an archive database using a minimal amount of traditional disk storage. When you load an archive database, the pages residing in the database dump are not copied by the Backup Server. Instead, Adaptive Server creates a map that represents a "logical-to-virtual" mapping of the pages within the archive. This significantly decreases the amount of time required to view the data in a database dump, and reduces the storage requirement for loading the dump.

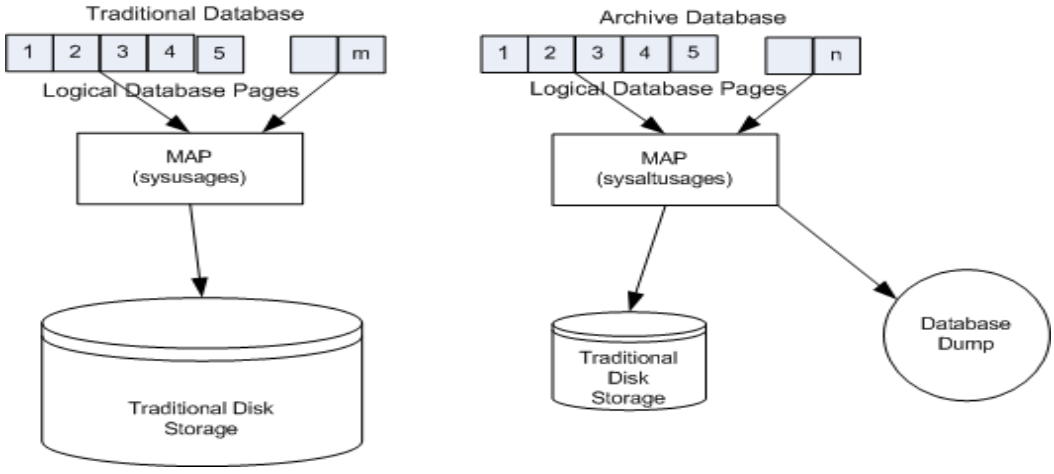
An archive database does not have to be a complete copy of the original database. Depending on the optimization used when dumping the database using sp_dumpoptimize, an archive database may be fully populated (every page in the database is in the database dump), or partially populated (only allocated pages are stored in the database dump).

Because the database dump is presented as a read-only database, a database administrator can query it using familiar tools and techniques such as:

- Running database consistency checks on the most recent copy of a dump made from a production database. These checks can be offloaded to a different server to avoid resource contention in the production environment. If resources are not a concern, the archive can be directly checked in the same server in which it was created. Verification on the archive provides the assurance needed prior to performing a restore operation.
- If the integrity of a database dump is in question, loading it into an archive database can be a quick test for success, and therefore a good tool to identify the appropriate database dump that should be used to restore a traditional database.
- Object-level restoration from the database dump. Lost data is recovered using select into to copy the to-be-restored rows from the table within the archive database. The select into operation can be performed either directly in the server hosting the archive database, or by using Component Integration Services proxy tables if the archive database is available on a different server than that of the object requiring restoration.

In addition, transaction logs can be loaded into an archive database thereby providing the assurance that the same load sequence can be applied when performing a restore operation. Figure 4-1 represents the differences between an archive database and a traditional database structure.

Figure 4-1: Archive database components



Components of an archive database

An archive database is made up of three components working together to create the illusion that a database dump is functioning as a traditional database:

- The database dump (the archive)
- Optional traditional disk storage used to store the modified pages section
- The scratch database that hosts the sysaltusages table

The database dump

The read-only database dump is used as a repository for most unmodified pages.

You cannot make changes to the database dump. Any changes you make to the data within the dump are stored in the modified pages section.

Adaptive Server sees the database dump and its stripes as database devices that are usable only by the archive database.

The modified pages section

Database dumps reflect a snapshot of a database at a given moment in time. The archive database that represents a database dump is read-only. No user transactions are allowed. Although the archive database is read-only, some modifications are allowed. For example:

- You can run recovery to make the archive database consistent with the source database.
- dbcc commands that perform fixes are allowed so that fixed versions of tables can be restored.

These modified and newly-allocated database pages cannot be stored within the database dump and its stripes, therefore an archive database requires some traditional database storage. This disk space is referred to as the modified pages section, and you can allocate it using the create archive database and alter database commands.

The modified pages section is divided into two segments:

- The disposable changes segment stores any page that is modified or allocated by the recovery undo pass or any log page that is modified or allocated at any time. A page has only one entry in the disposable changes section.
- The permanent changes segment stores any other page modification or allocation. A page has only one entry in the permanent changes section.

When you resize the modified pages section, sysusages rows are updated in the master database.

The sysaltusages table and the scratch database

The *sysaltusages* table

The sysaltusages system table is a new data-only-locked table that is used to map page numbers in an archive database to the actual page within either the database dump and its stripes, or the modified pages section. However, unlike the sysusages table in a traditional database, the sysaltusages table does not map every logical page in the database. sysaltusages maps:

- Pages that have been stored in a database dump
- Pages that have been modified, and therefore, relocated to the modified pages section

sysaltusages has the following columns:

```
dbid    location    lstart    start    size    vstart    vdevno    segmap
```

where:

- dbid – is the database ID of the archive database.
- location – is the location of the archive database segment where the physically contiguous block of pages resides.

  In the location column, a value of 5 and 6 means the location is in the database dump, transaction log dump, or their stripes, and a value of 7 or 8 means that the location is in the modified pages section. A value of 4 is used to fill the gaps for pages that are not physically available.
- lstart – is the logical page number of the start of the block of physically contiguous pages.
- size – is the number of logical pages in the block of physically contiguous pages.
- vstart – is the offset of the start of the contiguous block of pages on the device given by vdevno.
- vdevno – is the device number on which the contiguous block of pages resides.
- segmap – is a map of the segments to which this block of pages is allocated.

Note Because sysaltusages is a row-locked catalog, you may need to periodically use reorg to reclaim logically deleted space.

The scratch database

The scratch database stores the new sysaltusages table. The scratch database is used to provide flexibility as to where the sysaltusages table is located.

The scratch database can be any database (with some exceptions like master and temporary databases.) Sybase recommends that you dedicate a database that is used only as a scratch database, because:

- The size of sysaltusages may vary depending on the number of archive databases it supports. You cannot decrease the size of a database, but if it is too large, you can drop it and re-create a smaller database when required.
- It allows you to turn on the "trunc log on checkpoint" option so that the database log be automatically truncated.

Apart from hosting the sysaltusages table, this database is like any other. You can use threshold procedures and other space management mechanisms to manage space within the database.

You must specify a database that is to be used as a scratch database, by entering:

```
sp_dboption <db name>, "scratch database", "true"
```

Each archive database can be assigned to only one scratch database at a time, but multiple archive databases can use the same scratch database. If you have a large number of archive databases, you may want to define multiple scratch databases.

Working with an archive database

You can perform many traditional database operations on an archive database. However, user-defined transactions and commands that modify the database such as such as insert, update, and delete are not allowed.

A populated archive database is similar to a read-only database where the readonly option has been applied using sp_dboption.

Configuring an archive database

This section provides information about how to create and configure an archive database.

Creating an archive database

Create an archive database by issuing the create archive database command:

```
create archive database <db name>
    [on <db device> [= <size>]
        [, <db device> [= <size>] ] ... ]
    with scratch_database = <db name>
```

where:

- on – specifies the modified pages section. Adaptive Server requires traditional database storage to store modified pages. Use the on clause to specify the location and size of the modified pages section.
  - *db device* – specifies the database device on which you want to create your modified pages section.

- *size* – specifies the size of the modified pages section you want to create. If you omit *size*, 5120 pages are allocated.

- with scratch_database – (required) specifies the name of an existing database in which information about the archive database is maintained. The sysaltusages system table, which maps logical pages in the archive database onto physical pages, is stored in the scratch database.

Sizing the modified pages section

The modified pages section is used to store database pages that are modified or newly allocated. These pages are stored in either the permanent changes segment, the disposable changes segment, or both.

- A page can be remapped to the permanent changes section only once.
- A page can be remapped to the disposable changes section only once.
- Recovery is responsible for most page remappings.
- dbcc checkalloc also requires significant space allocation.
- You can increase the size of the modified pages section using the alter database command. However, to decrease the size of the modified pages section, you must drop the archive database and re-create it.

The permanent and disposable changes segments are logically distinct. The permanent changes segment is defined by the set of sysusages fragments that contain a segmap to the permanent changes segment. The disposable changes segment is defined by the set of sysusages fragments containing a segmap to the disposable changes section. The disposable changes segment is discarded at the beginning of each load tran command.

Note Adaptive Server automatically manages the division of space in the modified pages section between the permanent changes segment and the disposable changes segment. When this automatic resizing is done, sysusages rows are updated in the master database.

The minimum size of the modified pages section depends on how many pages are modified or newly allocated in the database. Many of these pages are modified by redo recovery and undo recovery.

Use the load database with norecovery command to minimize the number of modified pages, and therefore, the amount of space required in the modified pages section. There are downsides to doing this. For more information, see "Using load database with norecovery" on page 84.

Note dbcc checkalloc consumes a large amount of space in the modified pages section, even if you use the nofix option. When you run dbcc checkalloc, every allocation page (every 256th page) has information written to it. These allocation-page modifications are stored in the modified pages section, and means that when you are using dbcc checkalloc, you need a modified pages section that is at least 1/256th the size of the original database.

If you do not have enough space in the modified pages section, the command that requires the space is suspended and you see an error similar to:

```
There is no more space in the modified pages section for
the archive database <database name>. Use the ALTER
DATABASE command to increase the amount of space
available to the database.
```

To increase space in the modified pages section, either:

- Use alter database to increase the size of the modified pages section, or
- If you do not want to allocate additional space to the modified pages section, enter Ctrl+C to abort the current command.

Note You cannot use thresholds to manage the space in the modified pages section.

Increasing the amount of space allocated to the modified pages section

You can use alter database to add space to the modified pages section of the archive database. Increasing the space in the modified pages section allows a suspended command to resume operation.

You can use alter database at any time to increase the size of the modified pages section, not only when space runs out.

```
alter database <db name>
    [ on <db device> [= <size>]
        [, <db device> {= <size>] ] ...]
```

Materializing an archive database

An archive database is a placeholder that is useful only once it has been loaded with a database dump. The load process does not actually copy pages, however, it materializes the database using page mapping.

Use the load database command to materialize an archive database:

```
load database <db name>
    from <dump device>
    [ [stripe on <stripe device>] ... ]
    [with [norecovery,][passwd=<password>]
```

where:

- *db name* – specifies the name of the archive database into which you want to load.
- *dump device* – specifies the name of the disk database dump from which you want to load the dump.
- *stripe device* – specifies additional disk database dump stripes.
- norecovery – indicates that the load database command will not run recovery, and that the database is brought online automatically after the load database command has completed. For more information, see "Using load database with norecovery" on page 84.

  Note norecovery was introduced in Adaptive Server version 12.5.4 and 15.0.2 for archive database access. You cannot use norecovery on a traditional database.

- passwd =<*password*> – must be specified if the dump from which you are loading the archive database is password-protected. The password must be between 6 and 30 characters long.

Note You do not need to have Backup Server running when loading a database dump into an archive database.

Using *load database with norecovery*

The with norecovery option of the load database command allows a database dump to be loaded into an archive database without recovering anything, reducing the time required to load. Many database pages can be modified or allocated during recovery, causing them to be stored in the modified pages section. Therefore, skipping recovery consumes minimum space in the modified pages section. The with norecovery option allows a quick view into an archive database.

If you use with norecovery, the database is brought online automatically.

However, using load database with norecovery for a database that requires recovery may leave it transactionally and physically inconsistent. Running dbcc checks on a physically inconsistent database may produce many errors.

Once you have loaded an archive database with norecovery, you must have sa_role or database owner privileges to use it.

Using logical devices with an archive database

You can use sp_addumpdevice to create a logical device from which an archive database can be loaded:

```
sp_addumpdevice 'archive database', '<logical name>',
'<physical name>'
```

After you have executed this command, use the *logical name* instead of the *physical name* as the *dump device* or *stripe device* in a load database command.

Note You cannot use an archive database logical device as a device specification for a load into a traditional database or when dumping a traditional database.

---

load database limitations with an archive database

load database has the following limitations when used with an archive database:

- The database dump for an archive database is required to be a disk dump on a file system mounted on the local machine. This can be local storage or NFS storage. `load database ... at <remote server>` syntax is not supported, nor are database dumps on tape.

- Cross-architecture loads are not supported. The database dump and the load database command must be performed on the same architecture with respect to byte ordering.
- The dumped database must have the same page size as that used by the server that is hosting the archive database.
- The major version of the server on which the dump was taken must be earlier than or equal to the major version of the server hosting the archive database.
- The character set and sort order on the server on which the database dump was taken must be the same as the character set and sort order of the server hosting the archive database.

Bringing an archive database online

To bring an archive database online, use:

```
online database <db name>
```

where *db name* is the archive database you want to bring online.

online database performs undo recovery during which modified and allocated pages may be remapped to the modified pages section.

You need not bring a database online if it has been loaded with norecovery, since the load automatically brings the database online without running the recovery undo pass.

Loading a transaction log into an archive database

To load a transaction log into an archive database, use:

```
load tran <db name>
    from <dump device>
    [ [stripe on <stripe device>] ... ]
```

where:

- *db name* – is the archive database into which you are loading the transaction log.
- *dump device* – is a local disk transaction log dump.
- *stripe device* – specifies the stripes for the local disk transaction log dump.

When you load a transaction log into an archive database, load tran runs the recovery redo pass. Modified and new database pages are written to the permanent changes segment. You must have enough space in the modified pages section to accommodate these changes. If necessary, increase space for the modified pages section by using alter database to increase the normal database storage allocated to the archive database.

Unlike a traditional database, an archive database can be brought online in the middle of a load sequence without breaking the load sequence. When a traditional database is loaded and then brought online without using the for standby_access clause, it is no longer possible to load the next transaction log in the load sequence. An archive database however, can be brought online without the for standby_access clause and later, loaded with the next transaction log in the load sequence. This allows read-only operations like running consistency checks, at any time during the load sequence. This is possible because when loading a transaction log into the archive database, Adaptive Server automatically removes the disposable changes segment from the modified pages section. This effectively reverts the archive database to its state after the previous load was done, thereby allowing the next transaction log in the sequence to be loaded.

Dropping an archive database

To drop an archive database, use:

drop database <*db name*>

where *db name* is the database you want to drop.

When dropping an archive database, all the rows for that database are deleted from the sysaltusages table in the scratch database. This requires log space in the scratch database.

Using an archive database

This section provides information on the commands you run on an archive database.

Using SQL commands with an archive database

In addition to the commands already documented, (alter database, load database, online database, drop database and load tran) the following SQL commands are allowed in an archive database:

- use
- select
- select into – where the target database is not an archive database.
- Cursor operations that perform reads, including:
  - declare cursor
  - deallocate cursor
  - open
  - fetch

  You cannot use an updatable cursor.
- checkpoint – is a supported command. However, the checkpoint process does not checkpoint an archive database automatically.
- execute is allowed as long as any statements that reference the archive database are allowed within the archive database. A transaction inside or outside a stored procedure is not permitted with an execute command.
- lock table
- readtext

Note DML commands including insert, update, and delete are not permitted, and you cannot start user transactions.

Using dbcc commands with an archive database

The following dbcc commands are allowed in an archive database:

- checkdb

- checkcatalog

  Note The fix version of checkcatalog is not supported.

- checktable
- checkindex
- checkalloc
- checkstorage
- indexalloc
- tablealloc
- textalloc

While dbcc commands are executing, other users cannot access an archive database. If you attempt to access an archive database while dbcc commands are being performed, you receive a message saying that the database is in single-user mode.

You can use variants of the above dbcc commands on an archive database that is online or offline. However, you can use a dbcc command with a fix option only on an archive database that is online.

Typical archive database command sequence

The following syntax could be a typical archive database command sequence.

First, create the scratch database if necessary, using the create database command.

```
create database scratchdb
      on datadev1 = 100
      log on logdev1 = 50
```

This creates a 150MB traditional database called scratchdb.

Use sp_dboption to designate the database you just created as a scratch database:

```
sp_dboption "scratchdb", "scratch database", "true"
```

Create the archive database.

```
create archive database archivedb
      on datadev2 = 20
```

```
      with scratch_database = scratchdb
```

This creates an archive database called archivedb, with a 20MB modified pages section.

Materialize your archive database using load database:

```
load database archivedb
      from "/dev/dumps/050615/proddb_01.dmp"
      stripe on "/dev/dumps/050615/proddb_02.dmp"
```

Bring the database online:

```
online database archivedb
```

Check the consistency of the archive database using dbcc commands. For example:

```
dbcc checkdb(archivedb)
```

Load a transaction log dump using load tran and restore objects from the archive database using select into or bcp.

```
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_01.dmp"
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_02.dmp"
online database archivedb
select * into proddb.dbo.orders from
      archivedb.dbo.orders
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_03.dmp"
online database archivedb
```

Security and permissions for an archive database

Permission to execute commands and stored procedures, and access to objects in an archive database is the same as for a traditional database loaded with the same database dump on the same server.

When an archive database is loaded with norecovery, access to that database is limited to users with sa_role, or the database owner.

Compressed dumps for an archive database

To use a compressed dump for an archive database, you must:

- Create the compressed dump with the with compression = <compression level> option of the dump database or dump tran command.
- Create a memory pool for accessing the archive database.

Note Dumps generated with "compress::" cannot be loaded into an archive database. Therefore, any references to compression in this chapter refer to dumps generated using the with compression = <compression level> option.

Compression logic has changed in version 15.0 ESD #2. See "Compatibility issues for a compressed dump" on page 92.

Creating a compression memory pool

When Adaptive Server reads a page from a compressed dump, it selects a compressed block from the dump, decompresses it, and extracts the required page. The decompression in Adaptive Server is done using larger buffers from a special memory pool. The size of the pool is configured using:

```
sp_configure 'compression memory size', <size>
```

This is a dynamic configuration parameter, and the size is given in 2KB pages. If size is set to 0, no pool is created and a compressed dump cannot be loaded.

To determine the optimal size for your pool, consider these two factors:

- The block I/O used by the Backup Server. By default, this block I/O is 64KB, but it could have been changed using the with blocksize option in the dump database command.
- The number of *concurrent* users decompressing blocks within all archive databases. Each concurrent user requires two buffers each the same size as the block I/O.

As an absolute minimum, allow one concurrent user two buffers per archive database.

Migrating an archive database

sybmigrate does not migrate an archive database if an entire installation is being migrated.

sybmigrate migrates an archive database only if the archive database is specifically selected for migration. When you migrate an archive database to a target server, sybmigrate automatically creates a traditional database—rather than an archive database—on the target server.

Upgrading and downgrading an archive database

This section describes how to upgrade and downgrade an archive database.

Upgrading an Adaptive Server with an archive database

You cannot upgrade an archive database. If you load a database dump from an older version of Adaptive Server onto an archive database hosted on a newer version of Adaptive Server, the database is not internally upgraded when you execute online database.

If you upgrade an Adaptive Server containing an archive database, all the databases except the archive databases are upgraded. The archive database remains on the older version of Adaptive Server.

Sybase recommends you reload the archive database with a dump generated from an already upgraded database.

For more information about upgrading Adaptive Server, see the *Installation Guide* for your platform.

Downgrading an Adaptive Server with an archive database

When you are downgrading to a version of Adaptive Server that does not support archive databases, be aware of the following:

- If you must downgrade an Adaptive Server containing an archive database to a version of Adaptive Server that does not support archive databases, Sybase recommends you drop the archive database before you downgrade.

To eliminate the new sysaltusages table, drop the scratch database before you perform the downgrade procedure. sysaltuages does not cause any problems if the scratch database is not dropped.

- Backup Server versions 15.0 ESD #2 and later writes a new format for compression (with compression = <compression level>) so that the dump can be loaded into an archive database. Therefore, if you must load a compressed dump onto a version of Adaptive Server that does not support archive databases access, use the same version of Backup Server that created the compressed dump to load the compressed database dump. An earlier version of Backup Server does not support the new format of the compressed database dump.

  When you are downgrading without compression, you need not worry about Backup Server at all.

Compatibility issues for a compressed dump

- You cannot load dumps generated with "compress::" into an archive database. There are no compatibility issues with dumps using this compression option on traditional databases.
- The format of a compressed dump generated with the with compression = <compression level> option has changed. Backup Server versions 15.0 ESD #2 and later is the component that writes the new compression format. Therefore:
  - A compressed dump made using a Backup Server version 15.0 ESD #2 and later can be loaded only into a pre-15.0 ESD #2 installation using a Backup Server version 15.0 ESD #2 or later.
  - If you are using a pre-15.0 ESD #2 installation and want to use your dumps for an archive database, use Backup Server version 15.0 ESD #2 or higher to create compressed database dumps.

  Note A Backup Server version 15.0 ESD #2 and later understands both 15.0 ESD #2 and earlier compression formats; therefore, you can use a 15.0 ESD #2 Backup Server for both dump and loads.

DDLGen support for archive database access

To generate DDL for all archive databases, use the extended filter option "OA."

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -N% -XOA
```

To generate DDL for a single archive database, use the syntax for normal databases. This example creates DDL for the archive database archivedb:

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -Narchivedb
```

Archive database limitations

An archive database has the following limitations:

- An archive database is read-only.
- An archive database is automatically in single-user mode when any command is run that results in changes to the archive database, such as dbcc commands.
- An archive database uses only database dumps or transaction log dumps on disk. Tape dumps are not supported.
- The database dump or transaction log dumps must be visible from the server that is hosting the archive database. Remote dumps are not supported.
- For an archive database to access compressed dumps, the dump must have been created using the with compression option rather than the "compress::" option.
- The checkpoint process does not automatically checkpoint an archive database. Use the checkpoint command to checkpoint an archive database.
- You cannot use sp_dbrecovery_order to specify an archive database in the database recovery sequence. Archive databases are recovered last, in their *dbid* order.
- When pages are cached in an archive database, the cached pages stay in the memory pool with the same page size as the server. So, on a 2K server, the pages are always cached in a 2K pool. On a 16K server, the pages are always cached in a 16K pool.

- You cannot bind an archive database, or any object within that database, to a user-defined cache. Objects within an archive database default to the default data cache.
- disk resize does not work on any device used by an archive database and that maps to a database dump or a transaction log.
- disk refit does not rebuild the master database's sysusages entries from any devices that are used by an archive database. This applies both to dump devices and those used for the modified pages section. Existing sysusages entries for an archive database remain however.
- An archive database cannot be replicated.
- An archive database does not fail over on a high-availability server.
- Free-space thresholds cannot be established on an archive database.

CHAPTER 5

Statistical Aggregate Functions

In version 15.0.2, Adaptive Server introduces statistical aggregate functions to compute variance and standard deviation.

Overview

Aggregate functions summarize data over a group of rows from the database. The groups are formed using the group by clause of the select statement.

Simple aggregate functions, such as sum, avg, max, min, count_big, and count are allowed only in the select list and in the having and order by clauses as well as the compute clause of a select statement. These functions summarize data over a group of rows from the database.

Adaptive Server Enterprise now supports statistical aggregate functions, which permit statistical analysis of numeric data. These functions include stddev, stddev_samp, stddev_pop, variance, var_samp, and var_pop.

These functions, including stddev and variance, are true aggregate functions in that they can compute values for a group of rows as determined by the query's group by clause. As with other basic aggregate functions such as max or min, their computation ignores null values in the input. Also, regardless of the domain of the expression being analyzed, all variance and standard deviation computation uses IEEE double-precision floating-point standard.

If the input to any variance or standard deviation function is the empty set, then each function returns as its result a null value. If the input to any variance or standard deviation function is a single value, then each function returns 0 as its result.

Standard deviation and variance

The new statistical aggregate functions (and their aliases) are:

- stddev_pop (also stdevp) – standard deviation of a population. Computes the population standard deviation of the provided value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the square root of the population variance. See stddev_pop on page 102 for syntax and usage information.
- stddev_samp (also stdev, stddev) – standard deviation of a sample. Computes the population standard deviation of the provided value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the square root of the sample variance. See stddev_samp on page 104 for syntax and usage information.
- var_pop (also varp) – variance of a population. Computes the population variance of value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the sum of squares of the difference of value expression from the mean of value expression, divided by the number of rows in the group. See var_pop on page 107 for syntax and usage information.

- var_samp (also var, variance) – variance of a sample. Computes the sample variance of value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the sum of squares of the difference from the mean of the value expression, divided by one less than the number of rows in the group. See var_samp on page 109 for syntax and usage information.

Statistical aggregates

Statistical aggregates are similar to the avg aggregate in that:

- The syntax is:

  var_pop ([all | distinct] *expression*)

- Only expressions with numerical datatypes are valid.
- Null values do not participate in the calculation.
- The result is NULL only if no data participates in the calculation.
- The distinct or all clauses can precede the expression (the default is all).
- You can use statistical aggregates as vector aggregates (with group by), scalar aggregates (without group by), or in the compute clause.

Unlike the avg aggregate, however, the results are:

- Always of float datatype (that is, a double-precision floating-point), whereas for the avg aggregate, the datatype of the result is the same as that of the expression (with exceptions).
- 0.0 for a single data point.

Formulas

Figure 5-1: The formula for population-related statistical aggregate functions

The formula that defines the variance of the population of size *n* having mean μ (var_pop) is as follows. The population standard deviation (stddev_pop) is the positive square root of this.

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = Variance

n = Population size

μ = Mean of the values x_i

Figure 5-2: The formula for sample-related statistical aggregate functions

The formula that defines an unbiased estimate of the population variance from a sample of size *n* having mean $\bar{x}$ (var_samp) is as follows. The sample standard deviation (stddev_samp) is the positive square root of this.

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = Variance

n = Sample size

$\bar{x}$ = Mean of the values x_i

The essential difference between the two formulas is the division by *n-1* instead of *n*.

These two functions are similar, but are used for different purposes:

- var_samp – is used when you want evaluate a sample—that is, a subset—of a population as being representative of the entire population
- var_pop – is used when you have all of the data available for a population, or when *n* is so large that the difference between *n* and *n-1* is insignificant

stddev

Description

Computes the standard deviation of a sample consisting of a numeric expression, as a double.

Note stddev and stdev are aliases for stddev_samp. See stddev_samp on page 104 for details.

stdev

Description

Computes the standard deviation of a sample consisting of a numeric expression, as a double.

Note stddev and stdev are aliases for stddev_samp. See stddev_samp on page 104 for details.

stdevp

Description

Computes the standard deviation of a population consisting of a numeric expression, as a double.

Note stdevp is an alias for stddev_pop. See stddev_pop on page 102 for details.

stddev_pop

Description

Computes the standard deviation of a population consisting of a numeric expression, as a double. stdevp is an alias for stddev_pop, and uses the same syntax.

Syntax

stddev_pop ([all | distinct] *expression*)

Parameters

all
applies stddev_pop to all values. all is the default.

distinct
eliminates duplicate values before stddev_pop is applied.

expression
is the expression—commonly a column name—in which its population-based standard deviation is calculated over a set of rows.

Examples

The following statement lists the average and standard deviation of the advances for each type of book in the pubs2 database.

```
select type, avg(advance) as "avg", stddev_pop(advance)
  as "stddev" from titles group by type order by type
```

Usage

Computes the population standard deviation of the provided value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the square root of the population variance.

Figure 5-3: The formula for population-related statistical aggregate functions

The formula that defines the variance of the population of size *n* having mean μ (var_pop) is as follows. The population standard deviation (stddev_pop) is the positive square root of this.

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = Variance

n = Population size

μ = Mean of the values x_i

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Any user can execute stddev_pop.

See also

For general information about aggregate functions, see “Aggregate functions” in *Adaptive Server Enterprise Reference Manual: Building Blocks*.

Functions stddev_samp, var_pop, var_samp

stddev_samp

Description

Computes the standard deviation of a sample consisting of a numeric expression, as a double. stdev and stddev are aliases for stddev_samp, and use the same syntax.

Syntax

stddev_samp ([all | distinct] *expression*)

Parameters

all
applies stddev_samp to all values. all is the default.

distinct
eliminates duplicate values before stddev_samp is applied.

expression
is any numeric datatype (float, real, or double precision) expression.

Examples

The following statement lists the average and standard deviation of the advances for each type of book in the pubs2 database.

```
select type, avg(advance) as "avg",
  stddev_samp(advance) as "stddev" from titles
  where total_sales > 2000 group by type order by type
```

Usage

Computes the sample standard deviation of the provided value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the square root of the sample variance.

Figure 5-4: The formula for sample-related statistical aggregate functions

The formula that defines an unbiased estimate of the population variance from a sample of size n having mean $\bar{x}$ (var_samp) is as follows. The sample standard deviation (stddev_samp) is the positive square root of this.

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = Variance

n = Sample size

$\bar{x}$ = Mean of the values x_i

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Any user can execute stddev_samp.

See also

For general information about aggregate functions, see “Aggregate functions” in *Adaptive Server Enterprise Reference Manual: Building Blocks*.

Functions stddev_pop, var_pop, var_samp

var

Description

Computes the statistical variance of a sample consisting of a numeric expression, as a double, and returns the variance of a set of numbers.

Note var and variance are aliases of var_samp. See var_samp on page 109 for details.

var_pop

Description

Computes the statistical variance of a population consisting of a numeric expression, as a double. varp is an alias for var_pop, and uses the same syntax.

Syntax

var_pop ([all | distinct] *expression*)

Parameters

all
applies var_pop to all values. all is the default.

distinct
eliminates duplicate values before var_pop is applied.

expression
is an expression—commonly a column name—in which its population-based variance is calculated over a set of rows.

Examples

The following statement lists the average and variance of the advances for each type of book in the pubs2 database:

```
select type, avg(advance) as "avg", var_pop(advance)
  as "variance" from titles group by type order by type
```

Usage

Computes the population variance of the provided value expression evaluated for each row of the group (if distinct was specified, then each row that remains after duplicates have been eliminated), defined as the sum of squares of the difference of value expression, from the mean of value expression, divided by the number of rows in the group or partition.

Figure 5-5: The formula for population-related statistical aggregate functions

The formula that defines the variance of the population of size *n* having mean μ (var_pop) is as follows. The population standard deviation (stddev_pop) is the positive square root of this.

$$\sigma^2 = \frac{\sum (x_i - \mu)^2}{n}$$

σ^2 = Variance

n = Population size

μ = Mean of the values x_i

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Any user can execute var_pop.

See also

For general information about aggregate functions, see “Aggregate functions” in *Adaptive Server Enterprise Reference Manual: Building Blocks*.

Functions stddev_pop, stddev_samp, var_samp

var_samp

Description

Computes the statistical variance of a sample consisting of a numeric-expression, as a double, and returns the variance of a set of numbers. var and variance are aliases of var_samp, and use the same syntax.

Syntax

var_samp ([all | distinct] *expression*)

Parameters

all
applies var_samp to all values. all is the default.

distinct
eliminates duplicate values before var_samp is applied.

expression
is any numeric datatype (float, real, or double) expression.

Examples

The following statement lists the average and variance of the advances for each type of book in the pubs2 database:

```
select type, avg(advance) as "avg", var_samp(advance)
  as "variance" from titles where
  total_sales > 2000 group by type order by type
```

Usage

var_samp returns a result of double-precision floating-point datatype. If applied to the empty set, the result is NULL.

Figure 5-6: The formula for sample-related statistical aggregate functions

The formula that defines an unbiased estimate of the population variance from a sample of size *n* having mean $\bar{x}$ (var_samp) is as follows. The sample standard deviation (stddev_samp) is the positive square root of this.

$$s^2 = \frac{\sum (x_i - \bar{x})^2}{n - 1}$$

s^2 = Variance

n = Sample size

$\bar{x}$ = Mean of the values x_i

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Any user can execute var_samp.

See also

For general information about aggregate functions, see "Aggregate functions" in *Adaptive Server Enterprise Reference Manual: Building Blocks*.

Functions stddev_pop, stddev_samp, var_pop

variance

Description

Computes the statistical variance of a sample consisting of a numeric expression, as a double, and returns the variance of a set of numbers.

Note var and variance are aliases of var_samp. See var_samp on page 109 for details.

varp

Description

Computes the statistical variance of a population consisting of a numeric expression, as a double.

Note varp is an alias of var_pop. See var_pop on page 107 for details.

CHAPTER 6

Eager and Lazy Aggregation

This chapter discusses eager and lazy aggregation in Adaptive Server.

Overview

Aggregate processing is one of the most useful operations in DBMS environments. It summarizes large amounts of data with an aggregated value, including:

- The minimum, maximum, sum, or average value of a column in a specified set of rows
- The count of rows that match a condition
- Other statistical functions

In SQL, aggregate processing is performed using the aggregation functions min(), max(), count(), sum(), and avg(), and group by and having clauses. The SQL language implements two aggregate processing types, *vector aggregation* and *scalar aggregation*. A select-project-join (SPJ) query illustrates these two types of aggregate processing:

```
select r1, s1
from r, s
where r2 = s2
```

Vector aggregation

In vector aggregation, the SPJ result set is grouped on the group by clause expressions, and then the select clause aggregation functions are applied to each group. The query produces one result row per group:

```
select r1, sum (s1)
from r, s
```

```
where r2 = s2
group by r1
```

Scalar aggregation

In scalar aggregation, there is no group by clause and the entire SPJ result set is aggregated, as a single group, by the same select clause aggregate functions. The query produces a single result row:

```
select sum (s1)
from r, s
where r2 = s2
```

Eager aggregation

Eager aggregation transforms the internal representation of queries such as those discussed above, and processes them as if the aggregation is performed incrementally: first locally, over each table, producing intermediate aggregate results over smaller local subgroups, and then globally after the join, thus combining the local aggregation results to produce the final result set.

These queries, which return the same result set over any data set, are *derived table* SQL-level rewrites of the vector and scalar aggregation examples above. They illustrate the eager aggregation transformations that Adaptive Server performs on the internal representation of the queries.

Vector aggregation

```
select r1, sum(sum_s1 * count_r)
from
    (select
    r1,r2,
    count_r = count(*)
    from r
    group by r1,r2
    )gr
    ,
    (select
       s2,
       sum_s1 = sum(s1)
    from s
    group by s2
    )gs
where r2-s2
group by r1
```

Scalar aggregation

```
select sum(sum_s1 * count_r)
from
    (select
       r2,
```

```
        count_r = count(*)
    from r
    group by r2
    )gr
    ,
    (select
        s2,
        sum_s1 = sum(s1)
    from s
    group by s2
    )gs
where r2 = s2
```

Eager aggregation plans are generated and costed by the optimizer, and can be chosen as the best plan. This form of advanced query optimization can result in orders of magnitude of performance gain, relative to the original SQL query.

Aggregation and query processing

From the perspective of query processing (QP), aggregation can be both a costly operation, and an operation whose placement has an important impact on query performance.

- Aggregation generally computes an aggregated value. Vector aggregation is costly, compared to scalar aggregation, as rows must be grouped together to obtain the aggregated result over a group; this implies, in general, a reordering of the rows through sorting or hashing, both costly operations.
- Aggregating after applying cardinality-reducing operators (such as filters) to the input set reduces the cost of aggregation and can thus improve overall query performance.
- Aggregating before applying cardinality-increasing operators (such as joins and unions) to the input set reduces the cost of aggregation and can thus improve overall query performance.
- Aggregating early can reduce the costs of parent operators through reducing their input set cardinality, and can thus improve overall query performance.
- Aggregation can dramatically reduce the cardinality of the input set in the result set, when the grouping columns have relatively few distinct value combinations.

- Some properties of the aggregation's input set, as already grouped (for example, when the aggregation is ordered on the grouping columns), reduce the cost of vector aggregation; in scalar aggregation, rows ordered on the aggregated column allow computing a min or max without accessing each input row.
- Plan fragment physical properties have a big impact on aggregation cost.

The naive QP implementation of aggregation places the scalar or vector aggregate operator, as indicated by the SQL query, over the SPJ part of its query block. However, there are algebraic transformations that preserve the semantics of the query and allow aggregation at other places in the operators tree:

- Pushing the aggregation down toward the leaves, to aggregate early (called eager aggregation).
- Pulling the aggregation up toward the root, to aggregate late (called lazy aggregation).

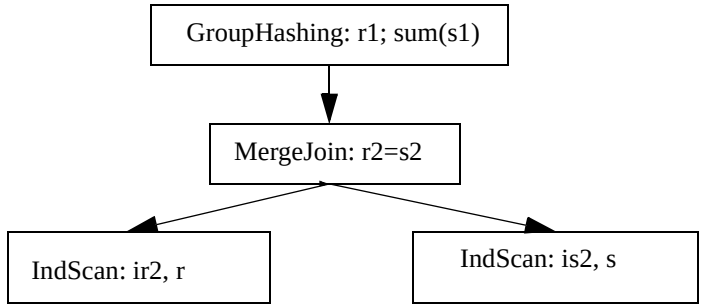
Plans obtained through such transformations differ greatly in performance. More importantly to distributed query processing (DQP), the cardinality of intermediate results can be greatly reduced by eager aggregation. Such orders-of-magnitude cardinality reduce cross-node data transfer cost, thus removing the main shortcoming of DQP as opposed to traditional QP.

Adaptive Server 15.0.2 implements eager aggregation over the leaves of a query plan, which means over the scan operators.

This query illustrates the QP implications of eager aggregation:

```
select r1, sum(s1)
from r,s
where r2 = s2
group by r1
```

Figure 6-1: Typical query execution plan

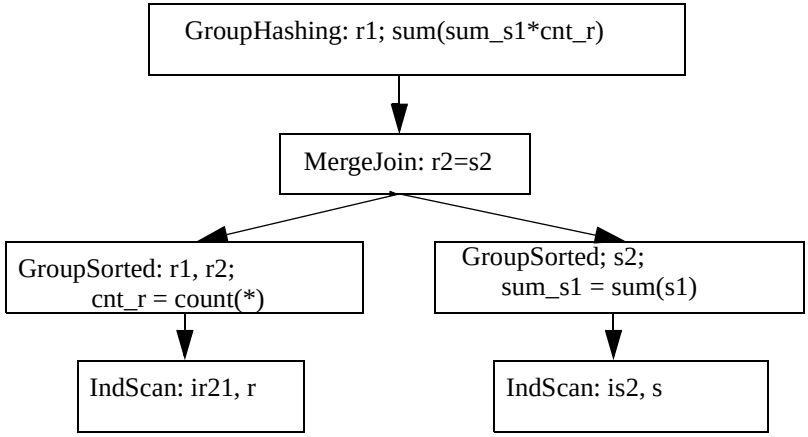


The two index scans, on r(r2) and s(s2) provide the orderings needed by the "r2=s2" merge join. Hash-based grouping is done over the join, as the query specifies it.

The optimizer also generates query plans that perform eager aggregation, also called the *push-down of grouping*, *early grouping*, or *eager grouping*. The SQL representation of the transform using derived tables is:

```
select r1, sum(sum_s1 * cnt_r)
from
    (select r1, r2, cnt_r = count(*)
          from r
          group by r1, r2
    ) as gr
    ,
    (select s2, sum_s1 = sum(s1)
          from s
          group by s2
    ) as gs
where r2 = s2
group by r1
```

Figure 6-2: Possible eager aggregation plan



The two eager GroupSorted operators group on the local grouping columns. GroupSorted operators apply to any column projected out for a reason other than that it is an aggregation function argument. These columns include:

- The main grouping columns in the group by clause
- Columns needed by predicates not yet applied

To place the cheap GroupSorted operator, the child plan fragment must provide ordering on all the local grouping columns; hence the `ir21` index on `r(r2, r1)`.

Examples

Dynamic archive

The most compelling reason to implement eager aggregation is dynamic archive, which is a distributed query processing (DQP) installation where recent OLTP read-write data is on an Adaptive Server and historical read-only data is on another server, either Adaptive Server or ASIQ.

The following view, `v`, offers decision support system (DSS) applications transparent access to local Adaptive Server data in ase_tab and, through the Component Integration Services (CIS) proxy_asiq_tab, to remote historical data on an ASIQ server.

```
create view v(v1, v2)
```

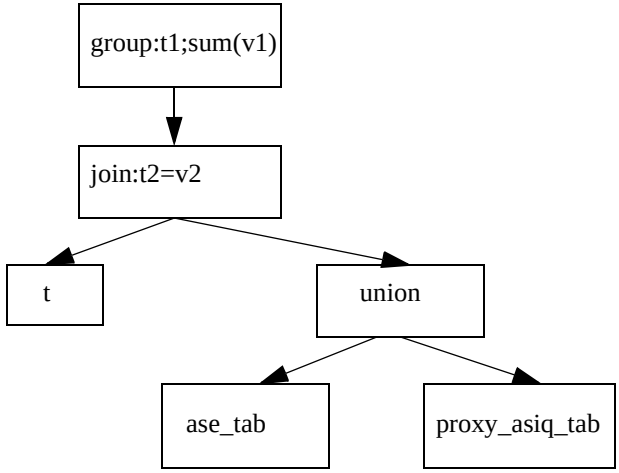
```
as
select a1, a2 from ase_tab
union all
select q1, q2 from proxy_asiq_tab
```

The DSS applications ignore the distributed nature of the data and use such union-in-view tables as the base tables of their complex queries, typically using aggregation:

```
select t1, sum(v1)
from t,v
where t2=v2
group by t1
```

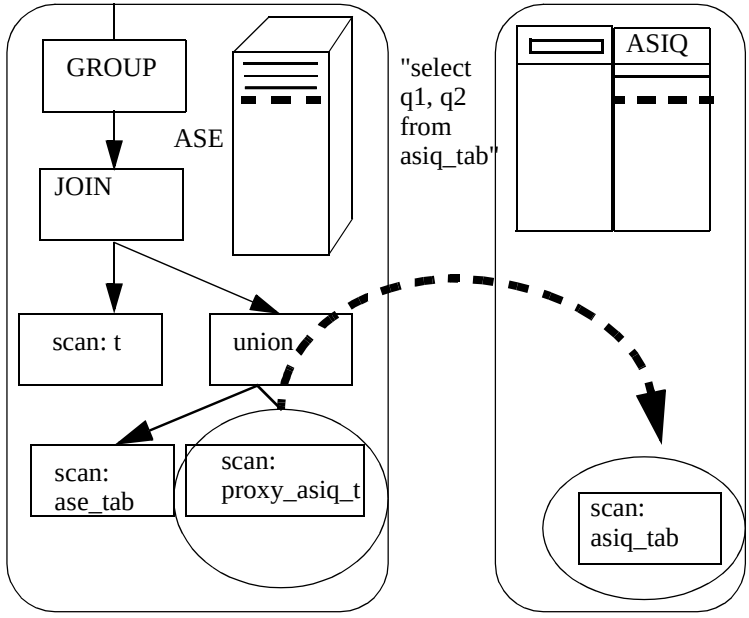
After view and union resolution, the following operator tree is obtained:

Figure 6-3: QL query rewrite



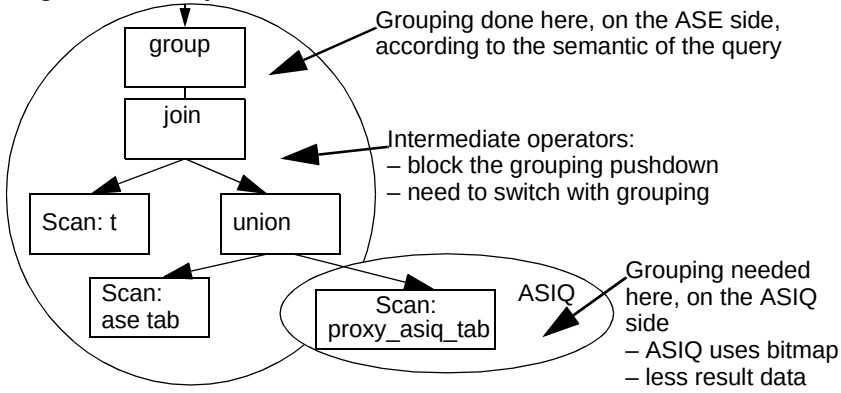
As this tree uses a CIS proxy table, the CIS layer uses a specialized remote scan operator to generate and ship a plan fragment to the remote site.

Figure 6-4: Suboptimal mechanism



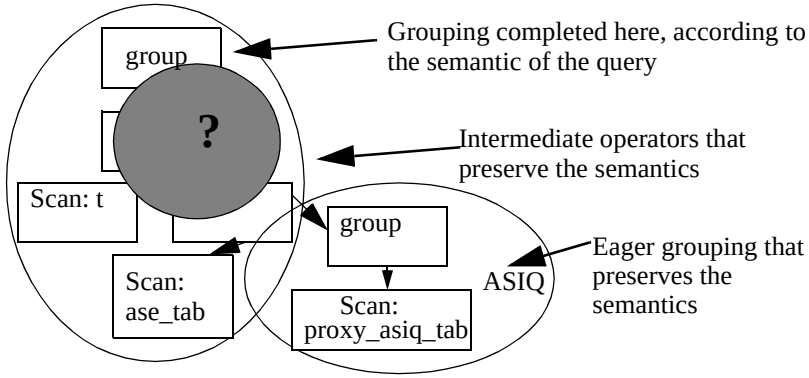
As such, this mechanism is suboptimal: the entire history table is shipped through the CIS layer to the Adaptive Server side, incurring a large network cost; furthermove, the advanced ASIQ bitmap-based grouping algorithms are not used.

Figure 6-5: Group here



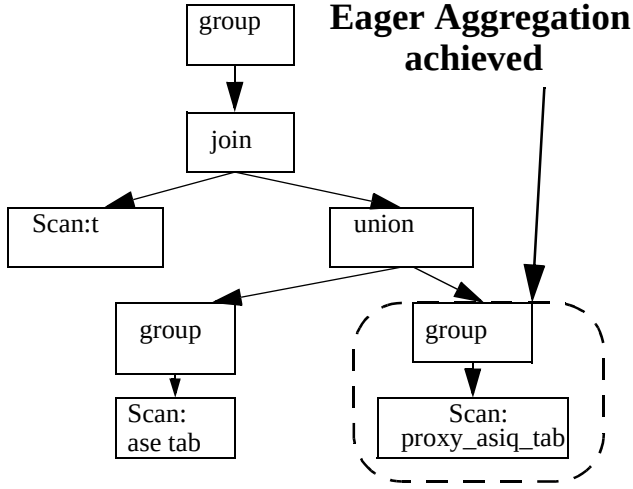
Ideally, transformations are performed on the operators tree and grouping on the ASIQ side, so that only aggregated data is transferred.

Figure 6-6: Grouping completed



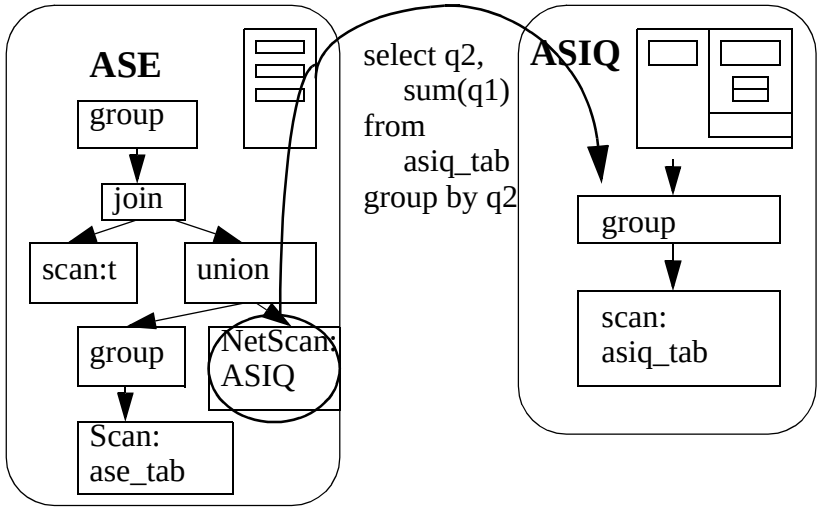
In this example, there are two operators between the group and the CIS proxy: a join and a union. The next transform pushes grouping below the join and the union, achieving *eager aggregation*:

Figure 6-7: Grouping done



Grouping is now adjacent to the CIS proxy. The CIS layer can now send a grouped query to ASIQ and return aggregated data.

Figure 6-8: Adjacent grouping



DSS/DQP

The efficient execution of complex aggregated DSS queries in a distributed environment is a challenge not only in dynamic archives; it is, in general, a generic DSS/DQP problem in dynamic archive DSS/DQP:

- A typical query involves complex joins and unions, and aggregation is performed at the top of the query tree.
- The data is distributed across the nodes, and the intermediate results must be shipped from a producer node to a consumer node for further processing.

Single-node DSS

Although the examples above are for DQP in general, and Dynamic Archive in particular, the eager aggregation performance impact goes beyond shipping intermediate results between DQP nodes.

Eager aggregation enhances the performance of aggregated complex queries by reducing intermediate result sets. Since aggregated complex queries are typical, eager aggregation enhances Adaptive Server performance in all DSS applications.

Using eager aggregation

Eager aggregation is an internal query processing feature. You need not change anything at the SQL level when you enable eager aggregation; queries that use aggregation have eager aggregation-based plans automatically enumerated and costed by the optimizer.

Enabling eager aggregation

Eager aggregation is controlled by the advanced_aggregation optimizer setting, which is off by default in all optimization goals except allrows_dss, where it is on. Eager aggregation can be enabled, disabled, or reset to the optimizer goal's default value at either the connection or query level.

For example, to enable at the connection level:

```
set advanced_aggregation on
```

To enable at the query level:

```
select r1, sum (s1)
from r, s
where r2 = s2
group by r1
plan
"(use advanced_aggregation on)"
```

Alternatively, if the optimization goal is set to allrows_dss, eager aggregation is implicitly enabled. In this example, an abstract plan sets allrows_dss at the query level:

```
select r1, sum (s1)
from r, s
where r2 = s2
group by r1
plan
"(use optgoal allrows_dss)"
```

Checking for eager aggregation

When eager aggregation is enabled, the optimizer determines cost, depending on whether the estimated cheapest plan uses eager aggregation or not.

Output from the showplan aggregation:

```
1> select r1, sum(s1)
2> from r, s
3> where r2=s2
4> group by r1
5> go

QUERY PLAN FOR STATEMENT 1 (at line 1).
STEP 1
The type of query is SELECT.
6 operator(s) under root
|ROOT:EMIT Operator
|
| |HASH VECTOR AGGREGATE Operator
| | GROUP BY
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable2 for internal storage.
| | Key Count: 1
| |
| | |MERGE JOIN Operator (Join Type: Inner Join)
| | | Using Worktable1 for internal storage.
| | | Key Count: 1
| | | Key Ordering: ASC
| | |
| | | |GROUP SORTED Operator
| | | | Evaluate Grouped COUNT AGGREGATE.
| | | |
| | | | |SCAN Operator
| | | | | FROM TABLE
| | | | | r
| | | | | Index : ir21
| | | | | Forward Scan.
| | | | | Positioning at index start.
| | | | | Index contains all needed columns. Base table
will not be read.
| | | | | Using I/O Size 2 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index
leaf pages.
| | | |
| | | | |GROUP SORTED Operator
| | | | | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | | | |
| | | | | |SCAN Operator
| | | | | | FROM TABLE
| | | | | | s
| | | | | | Index : is21
```

```
| | | | | Forward Scan.
| | | | | Positioning at index start.
| | | | | Index contains all needed columns. Base table
will not be read.
| | | | | Using I/O Size 2 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index
leaf pages.

r1
----------- -----------
          1           2
          2           4

(2 rows affected)
```

As the query performs vector aggregation over the join of r and s, the hash vector aggregate operator at the top of the query tree is expected in all cases. However, the group sorted operators over the scans of r and of s are not part of the query; they perform the eager aggregation.

When advanced_aggregation is off, the plan does not contain the eager aggregation operators group sorted:

```
1> set advanced_aggregation off
2> go
Ö
1> select r1, sum(s1)
2> from r, s
3> where r2=s2
4> group by r1
5> go
QUERY PLAN FOR STATEMENT 1 (at line 1).
STEP 1
The type of query is SELECT.
4 operator(s) under root
|ROOT:EMIT Operator
|
| |HASH VECTOR AGGREGATE Operator
| | GROUP BY
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable2 for internal storage.
| | Key Count: 1
| |
| | |MERGE JOIN Operator (Join Type: Inner Join)
| | | Using Worktable1 for internal storage.
| | | Key Count: 1
| | | Key Ordering: ASC
| | |
```

```
| | | |SCAN Operator
| | | | FROM TABLE
| | | | r
| | | | Index : ir21
| | | | Forward Scan.
| | | | Positioning at index start.
| | | | Index contains all needed columns. Base table
will not be read.
| | | | Using I/O Size 2 Kbytes for index leaf pages.
| | | | With LRU Buffer Replacement Strategy for index
leaf pages.
| | |
| | | |SCAN Operator
| | | | FROM TABLE
| | | | s
| | | | Index : is21
| | | | Forward Scan.
| | | | Positioning at index start.
| | | | Index contains all needed columns. Base table
will not be read.
| | | | Using I/O Size 2 Kbytes for index leaf pages.
| | | | With LRU Buffer Replacement Strategy for index
leaf pages.

r1
----------- -----------
          1           2
          2           4

(2 rows affected)
```

Forcing eager aggregation with abstract plans

The optimizer opportunistically enumerates the cheap GroupSorted-based eager aggregation plans when the child plan fragment provides an ordering on the local grouping columns.

This limitation avoids increasing the optimization search space and time. However, in some cases hash-based eager aggregation produces the cheapest plan. abstract plans can be used in such cases to force eager aggregation. advanced_grouping must be enabled to use such an abstract plan; otherwise the eager aggregation abstract plan is rejected.

In the example above, if r has no index on (r1, r2), and if r is large but has few r1--r2 distinct pairs of values, a hash join with eager grouping over r is the best plan, forced by this abstract plan:

```
1> select r1, sum(s1)
2> from r, s
3> where r2=s2
4> group by r1
5> plan
6> "(group_hashing
7>     (h_join
8>            (group_hashing
9>               (t_scan r)
10>           )
11>           (t_scan s)
12>    )
13> )"
14> go
QUERY PLAN FOR STATEMENT 1 (at line 1).
Optimized using the Abstract Plan in the PLAN clause.
STEP 1
The type of query is SELECT.
5 operator(s) under root
|ROOT:EMIT Operator
|
| |HASH VECTOR AGGREGATE Operator
| | GROUP BY
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable3 for internal storage.
| | Key Count: 1| |

| | |HASH JOIN Operator (Join Type: Inner Join)
| | | Using Worktable2 for internal storage.
| | | Key Count: 1
| | |
| | | |HASH VECTOR AGGREGATE Operator
| | | | GROUP BY
| | | | Evaluate Grouped COUNT AGGREGATE.
| | | | Using Worktable1 for internal storage.
| | | | Key Count: 2
| | | |
| | | | |SCAN Operator
| | | | | FROM TABLE
| | | | | r
| | | | | 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.
| | |
| | | |SCAN Operator
| | | | FROM TABLE
| | | | s
| | | | 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.
r1
----------- -----------
1           2
2           4

(2 rows affected)
```

T he hash vector aggregate operator eagerly aggregates the scan of r, as requested by the abstract plan.

CHAPTER 7

Changes that improve the performance for inserting data

Adaptive Server version 15.0.2 includes these changes to improve the performance of inserting data:

bcp performance optimization

In earlier versions, Adaptive Server used fast bcp when sp_dboption 'select into/bulkcopy/pllsort' was enabled and the table had no indexes or triggers.

In all other cases, Adaptive Server used slow bcp.

If a table had indexes or triggers, you had to drop these before using fast bcp and then recreate them after loading the data with bcp.

Adaptive Server version 15.0.2 allows you to use fast bcp to copy data into tables with non-clustered indexes or triggers, improving Adaptive Server's performance for inserting huge volumes of data. With this optimization the use of slow bcp is now only required when:

- sp_dboption 'select into/bulkcopy/pllsort' is off.
- sp_dboption 'select into/bulkcopy/pllsort' is on, but the table uses the allpages locking scheme and has a clustered index.
- sp_dboption 'select into/bulkcopy/pllsort' is on, but the table has a unique nonclustered index.

If the option ignore_dup_key option is enabled on the unique index, performing fast bcp can put the table and index in an inconsistent state if rows with duplicate keys are inserted. To avoid the inconsistency, Adaptive Server performs slow bcp.

- If the table has nonclustered indexes or triggers, and the table is marked for replication or the database is used as a warm standby.

  Because fast bcp does not log inserts, if Adaptive Server uses fast bcp, the rows bcp copies cannot be recovered on the replication site if there is a problem. Adaptive Server uses slow bcp in these situations to maintain compatibility with applications that were written to use the old behavior.

In all other cases, Adaptive Server uses fast bcp.

These are situations in which Adaptive Server version 15.0.2 uses fast bcp but earlier versions Adaptive Server used slow bcp (in all cases sp_dboption 'select into/bulkcopy/pllsort' is enabled and the table does not have a clustered index):

- When the table has a non-unique, nonclustered index. Adaptive Server logs the index updates and the page allocations only. It does not log inserts into the table.
- When a table has triggers. However, bcp does not fire any triggers in the target table.
- When a table has datarows or datapage locking scheme with a clustered index.

If the table includes nonclustered indexes or triggers, but sp_dboption 'select into/bulkcopy/pllsort' is not enabled, Adaptive Server uses slow bcp, and prints this warning message to indicate that you can improve the performance of bcp by enabling sp_dboption 'select into/bulkcopy/pllsort':

```
Performing slow bcp on table '%s'. To enable fast bcp
please turn on 'select into/bulkcopy' option on the
database '%s
```

bcp optimizatin is performed by Adaptive Server and does not require that you use Open Client version 15.0 or later.

Separate user log cache for a session's tempdb

Previous versions of Adaptive Server flushed the user log cache (ULC) as user sessions switched between transactions in the user databases and tempdb database. However, flushing the ULC causes logical IOs in syslogs and disk IOs, causing performance degradation.

Adaptive Server version 15.0.2 includes a separate ULC for the session's temporary database, so multi-database transactions that include a single user database and the session's temporaroy database do not require ULC flushes when the users switch between the databases or if all of the following conditions are met:

- Adaptive Server is currently committing the transaction.
- All the log records are in the ULC
- There are no post-commit log records.

Adaptive Server version 15.0.2 adds the configuration option, session tempdb log cache size, which allows you to configure the size of the ULC, helping to determine how often it needs flushing.

Summary information	
Default value	The logical page size
Range of values	The logical page size up to 2147483647
Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	

Using asynchronous writes during a page split

When Adaptive Server splits an index or data page, it moves some rows from the original pages to the newly created page. The operation of moving the rows is not logged. To ensure consistency and durability, Adaptive Server must satisfy these conditions:

- Adaptive Server writes the new page to disk before writing the modified page (with the rows removed) to disk. This ensures that Adaptive Server can restore the previous version of the page if the transaction is undone. Adaptive Server can find these rows on the new page and move them back to the old page even if the rows are missing in the old page, and their row contents were not logged.
- The new page reaches the disk before the transaction commits, which ensures that Adaptive Server cannot loose the committed data. If the transaction was committed, Adaptive Server is not required to redo the transaction for the new page, which would be impossible since the movement of the rows was not logged. In the case of undo, the new page's allocation is backed out; there's no page pre-image to restore.

Previous versions of Adaptive Server ensured these two conditions were met by synchronously writing the new page to disk. However, because the server could block until the synchronous write returned, this caused a degradation in performance.

Adaptive Server version 15.0.2 uses asynchronous writes to disk that satisfy the conditions described above and do not require the server to block as it waits for the write to complete.

Adaptive Server version 15.0.2 uses these asynchronous writes automatically and requires no configuration on your part.

Optimizations to improve throughput of tempdb transactions

Earlier versions of Adaptive Server flushed the data pages and single log records (SLRs) because crash recovery was not supported for tempdb or any databases not requiring recovery.

SLRs are log records that force a flush of the user log cache (ULC) to syslogs immediately after the record is logged. SLRs are created for OAM modifications, and Adaptive Server creates log records affecting allocation pages in a mixed log and data database as SLRs.

- For regular databases, a ULC containing SLRs is flushed immediately to avoid any undetected deadlocks caused during buffer "pinning". Avoiding a ULC flush for SLRs reduces log semaphore contention, improving the performance.

A ULC flush avoids the deadlock caused by buffer pinning. Because Adaptive Server does not pin the buffers for databases that do not need recovery, it avoids this deadlock and does not have to flush the ULC for SLRs.

- For databases that require recovery, Adaptive Server flushes dirty pages to disk during the checkpoint. This ensures that, if Adaptive Server crashes, all the committed data is saved to disk. However, for databases which do not require recovery, Adaptive Server supports a runtime rollback, but not a crash recovery. This allows it to avoid flushing dirty data pages at a checkpoint and improves performance.
- Adaptive Server does not support write ahead logging on databases that do not require recovery. Write-ahead logging guarantees that data for committed transactions can be recovered by "redoing" the log (reperforming the transactions listed in the log), and "undoing" the changes done by aborted or rolled back transactions to maintain database consistency. Write-ahead logging is implemented by the "buffer pinning" mechanism. Since Adaptive Server does not ensure write-ahead logging on databases not needing recovery, it does not pin buffers for these databases, so it can skip flushing the log when it commits a transaction.

Post-commit optimization

Previous versions of Adaptive Server:

- Used three scans of the log record after a committed transaction, one each for page deallocation, unreserved pages, and log deallocation.
- Performed the backward scan of log pages using page linkages. Some of the pages did not have log records requiring post-commit work.

Adaptive Server version 15.0.2 performs two scans of the log: the first scan looks for data page deallocation and unreserved pages, the second scan looks for log page deallocation. These scans are an internal optimization, transparent to users, and are performed automatically; you cannot switch the scans on or off.

With post-commit optimization, Adaptive Server remembers the “next” log page (in the backward direction) containing these log records. During the post-commit phase, Adaptive Server moves to the “next” page requiring post-commit work after processing records from a page. In a concurrent environment, where many users log their transactions to syslogs at the same time, post-commit optimization can improve the performance of post commit operation by avoiding reads or scans of unnecessary log pages.

The optimization does not show up in any diagnostics.

CHAPTER 8

Changes to the Query Processor

This chapter describes behavior changes that can affect Adaptive Server's query plan selection.

Deferred compilation

The query processor for Adaptive Server version 15.0.2 defers the optimization of statements in a stored procedure until it executes the statement. This benefits the query processor because the values for local variables are available for optimization for their respective statements.

Earlier versions of Adaptive Server used default guesses for selectivity estimates on predicates using local variables.

Non-binary character set histogram interpolation

In earlier versions of Adaptive Server, only the default binary character set benefited from histogram interpolation, which is used to estimate the selectivity of range predicates. For all other character sets, Adaptive Server made a selectivity estimate of 50 percent for a histogram cell. This typically required Adaptive Server to use a large number of histogram cells for character column histograms to reduce the error associated with this estimate.

Adaptive Server version 15.0.2 allows selectivity estimates to have the same accuracy as the binary character set, without requiring an excessive number of histogram steps. This benefits queries like the following with range predicates:

```
select * from t1 where charcolumn > "LMC0021" and
charcolumn <= "LMC0029"
```

If ranges specified falls into the same histogram cell, Adaptive Server can much more accurately estimate this selectivity.

Expression histogramming selectivity estimates

Earlier versions of Adaptive Server used default “guesses” for selectivity estimates.

Adaptive Server version 15.0.2 applies histogramming estimates to single column predicates if the histogram exists on the column. This results in more accurate row estimates, and improves the join order selection for query plans.

In this example, if the expression is very selective, it may be better to place table t1 at the beginning of the join order:

```
select * from t1,t2 where substring(t1.charcol, 1, 3)
= "LMC" and t1.a1 = t2.b
```

CHAPTER 9

Viewing Current Optimizer Settings

Adaptive Server version 15.0.2 introduces a new stored procedure that allows you to view current optimizer settings.

Overview

To give you the ability to easily view optimizer settings, Adaptive Server version 15.0.2 introduces a new fake table and a new stored procedure.

sysoptions is the new fake table. It stores information about each set option, its category, it's current and default settings. sysoptions also contains a bit map containing further detailed information about the option.

sp_options is a new stored procedure used to show option values.

sysoptions

sysoptions is the new fake table queried by sp_options. Table 9-1includes the name, datatype, attribute, and a brief description of each column in sysoptions. When you are querying sysoptions, the names of the rows are case sensitive.

Table 9-1: sysoptions

Name	Datatype	Attributes	Description
spid	int		Contains the process ID.

Name	Datatype	Attributes	Description
name	varchar(100)		Contains the name of the option.
category	varchar(100)		Contains the name of the category to which the option belongs.
currentsetting	varchar(100)	NULL	Contains the current setting of the option.
defaultsetting	varchar(100)	NULL	Contains the default setting of the option.
scope	int		Contains the bitmap used to capture information about options. The bits are ordered as follows: • Bit 1 – compiled time options • Bit 2 – stored procedure specific options • Bit 3 – binary options

Query sysoptions using sp_options. The datatype for the current and default value is varchar so settings with varchar values can be used directly. Settings with integer values can be used after typecasting.

You do not need special privileges to query sysoptions. For example:

```
select * from sysoptions
where spid = 13
go
```

You can also use string manipulation or typecasting. For example, if an option is numeric, you can query sysoptions by entering:

```
if (isnumeric(currentsetting))
    select@int_val = convert(int, currentsetting)
    ...
else
    select@char_val = currentsetting
    ...
```

sp_options

The syntax for sp_options is:

```
sp_options [ [show | help
                    [, <option_name> | <category_name> |null
                        [, dflt | non_dflt | null
                            [, <spid>] ] ] ] ]
```

where:

- show – lists the current and default values of all options, grouped according to their category. Issuing sp_options show with an option name specified gives you the current and default value for the individual option. You can also specify a session ID, and whether you want to view options with default settings or options with non-default settings.
- help – indicates that you wish to show usage information. You achieve the same result when you issue sp_options with no parameters.
- null – indicates the option for which you want to view the settings.
- *dflt* | *non_dflt* | null – indicates whether to show options with default settings or to show options with non-default settings.
- *spid* – specifies the session ID. Use the session ID to view other session settings.

Usage

Example 1 To view sp_options usage:

```
1> sp_options
2> go

Usage:
sp_options [ [show | help
            [, <option_name>|<category_name>|null
               [, dflt | non_dflt | null
                    [, <spid>] ] ] ] ]
```

Example 2 To view a list of all current and default options:

```
1> sp_options show
2> go
Category: Query Tuning
name                      currentsetting       defaultsetting       scope
----------------------------------------- ------------------------------
optgoal                     allrows_mix          allrows_mix          0
opttimeoutlimit             40                   10                   0
merge_join                  1                    1                    4
hash_join                   0                    0                    4
nl_join                     1                    1                    4
distinct_sorted             1                    1                    4
distinct_sorting            1                    1                    4
distinct_hashing            1                    1                    4
group_sorted                1                    1                    4
group_hashing               1                    1                    4
group_inserting             0                    0                    4
order_sorting               1                    1                    4
append_union_all            1                    1                    4
```

```
merge_union_all             1                   1                   4
merge_union_distinct        1                   1                   4
hash_union_distinct         1                   1                   4
store_index                 1                   1                   4
bushy_space_search          0                   0                   4
parallel_query              1                   1                   4
replicated_partition        0                   0                   4
ase125_primed               0                   0                   4
index_intersection          0                   0                   4
index_union                 1                   1                   4
multi_table_store_ind       0                   0                   4
advanced_aggregation        0                   0                   4
opportunistic_distinct_view 1                   1                   4
repartition_degree          3                   1                   2
scan_parallel_degree        0                   1                   2
resource_granularity        10                  10                  2
parallel_degree             0                   1                   2
statistics simulate         0                   0                   4
forceplan                   0                   0                   7
prefetch                    1                   1                   6
metrics_capture             0                   0                   6
process_limit_action        quiet               quiet               2
plan replace                0                   0                   4
plan exists check           0                   0                   4
plan dump                   0                   0                   4
plan load                   0                   0                   4

(39 rows affected)
(return status = 0)
```

Example 3 To view the current and default setting for an individual option:

```
1> sp_options show, "index_intersection"
2> go

name               category      currentsetting  defaultsetting  scope
-------------------------------------------------------------------------
index_intersection Query Tuning  0               0               4

(1 row affected)
(return status = 0)
```

Example 4 To show just the default setting for an individual option:

```
1> sp_options show, "index_intersection", dflt
2> go
```

```
name                    defaultsetting
-------------------------------------
index_intersection      0

(1 row affected)
(return status = 0)
```

Example 5 To show the current and default settings for a category:

```
1> sp_options show, "Query Tuning"
2> go

Category: Query Tuning

name                          currentsetting  defaultsetting  scope
-------------------------------------------------------------------
optgoal                       allrows_mix     allrows_mix     0
opttimeoutlimit               10              10              0
merge_join                    1               1               4
hash_join                     0               0               4
nl_join                       1               1               4
distinct_sorted               1               1               4
distinct_sorting              1               1               4
distinct_hashing              1               1               4
group_sorted                  1               1               4
group_hashing                 1               1               4
group_inserting               0               0               4
order_sorting                 1               1               4
append_union_all              1               1               4
merge_union_all               1               1               4
merge_union_distinct          1               1               4
hash_union_distinct           1               1               4
store_index                   1               1               4
bushy_space_search            0               0               4
parallel_query                1               1               4
replicated_partition          0               0               4
ase125_primed                 0               0               4
index_intersection            0               0               4
index_union                   1               1               4
multi_table_store_ind         0               0               4
advanced_aggregation          0               0               4
opportunistic_distinct_view   1               1               4
repartition_degree            3               1               2
scan_parallel_degree          0               1               2
resource_granularity          10              10              2
```

```
parallel_degree             0               1               2
statistics simulate         0               0               4
forceplan                   0               0               7
prefetch                    1               1               6
metrics_capture             0               0               6
process_limit_action        quiet           quiet           2
plan replace                0               0               4
plan exists check           0               0               4
plan dump                   0               0               4
plan load                   0               0               4

(39 rows affected)
(return status = 0)
```

Example 6 To show the default settings for the Query Tuning category:

```
1> sp_options show, "Query Tuning", dflt
2> go

Category: Query Tuning

name                          defaultsetting
-------------------------------------------
optgoal                       allrows_mix
opttimeoutlimit               10
merge_join                    1
hash_join                     0
nl_join                       1
distinct_sorted               1
distinct_sorting              1
distinct_hashing              1
group_sorted                  1
group_hashing                 1
group_inserting               0
order_sorting                 1
append_union_all              1
merge_union_all               1
merge_union_distinct          1
hash_union_distinct           1
store_index                   1
bushy_space_search            0
parallel_query                1
replicated_partition          0
ase125_primed                 0
index_intersection            0
index_union                   1
multi_table_store_ind         0
```

```
advanced_aggregation        0
opportunistic_distinct_view 1
repartition_degree          1
scan_parallel_degree        1
resource_granularity        10
parallel_degree             1
statistics simulate         0
forceplan                   0
prefetch                    1
metrics_capture             0
process_limit_action        quiet
plan replace                0
plan exists check           0
plan dump                   0
plan load                   0

(39 rows affected)
(return status = 0)
```

Example 7 To show the options set to a non-default setting in the Query Tuning category:

```
1> sp_options show, "Query Tuning", non_dflt
2> go

Category: Query Tuning

name                 currentsetting   defaultsetting
-------------------------------------------------
repartition_degree   3                1
scan_parallel_degree 0                1
parallel_degree      0                1

(3 rows affected)
(return status = 0)
```

Example 8 To show the options in the Query Tuning category:

```
1> sp_options, show, null
2> go

Category: Query Tuning

name                        currentsetting defaultsetting scope
------------------------------------------ -------------- -----
optgoal                     allrows_mix    allrows_mix        0
```

```
opttimeoutlimit             10              10              0
merge_join                  1               1               4
hash_join                   0               0               4
nl_join                     1               1               4
distinct_sorted             1               1               4
distinct_sorting            1               1               4
distinct_hashing            1               1               4
group_sorted                1               1               4
group_hashing               1               1               4
group_inserting             0               0               4
order_sorting               1               1               4
append_union_all            1               1               4
merge_union_all             1               1               4
merge_union_distinct        1               1               4
hash_union_distinct         1               1               4
store_index                 1               1               4
bushy_space_search          0               0               4
parallel_query              1               1               4
replicated_partition        0               0               4
ase125_primed               0               0               4
index_intersection          0               0               4
index_union                 1               1               4
multi_table_store_ind       0               0               4
advanced_aggregation        0               0               4
opportunistic_distinct_view 1               1               4
repartition_degree          3               1               2
scan_parallel_degree        0               1               2
resource_granularity        10              10              2
parallel_degree             0               1               2
statistics simulate         0               0               4
forceplan                   0               0               7
prefetch                    1               1               6
metrics_capture             0               0               6
process_limit_action        quiet           quiet           2
plan replace                0               0               4
plan exists check           0               0               4
plan dump                   0               0               4
plan load                   0               0               4
(39 rows affected)
(return status = 0)
```

Example 9 To show a list of the default settings for the Query Tuning category:

```
1> sp_options show, null, dflt
2> go
```

```
Category: Query Tuning

name                          defaultsetting
----------------------------- --------------
optgoal                       allrows_mix
opttimeoutlimit               10
merge_join                    1
hash_join                     0
nl_join                       1
distinct_sorted               1
distinct_sorting              1
distinct_hashing              1
group_sorted                  1
group_hashing                 1
group_inserting               0
order_sorting                 1
append_union_all              1
merge_union_all               1
merge_union_distinct          1
hash_union_distinct           1
store_index                   1
bushy_space_search            0
parallel_query                1
replicated_partition          0
ase125_primed                 0
index_intersection            0
index_union                   1
multi_table_store_ind         0
advanced_aggregation          0
opportunistic_distinct_view   1
repartition_degree            1
scan_parallel_degree          1
resource_granularity          10
parallel_degree               1
statistics simulate           0
forceplan                     0
prefetch                      1
metrics_capture               0
process_limit_action          quiet
plan replace                  0
plan exists check             0
plan dump                     0
plan load                     0

(39 rows affected)
(return status = 0)
```

Example 10 To show the options set to a non-default setting in the Query Tuning category:

```
1> sp_options show, null, non_dflt
2> go

Category: Query Tuning

name                 currentsetting defaultsetting
-------------------- -------------- --------------
repartition_degree   3              1
scan_parallel_degree 0              1
parallel_degree      0              1

(3 rows affected)
(return status = 0)
```

Example 11 If you enter a parameter that sp_options does not understand, you receive the following message:

```
1> sp_options show, "incorrect option"
2> go

Msg 19615, Level 16, State 1:
Procedure 'sp_options', Line 436:
No option or category matching 'incorrect option' is
found. Valid categories are:
category
------------
Query Tuning
(1 row affected)
(return status = 1)
```

Example 12 To see correct usage:

```
1> sp_options help
2> go

Usage:
sp_options [ [show | help
            [, <option_name>|<category_name>|null
               [, dflt | non_dflt | null
                    [, <spid>] ] ] ] ]
```

Use sp_options to view settings for the following options:

- set plan dump / load
- set plan exists check
- set forceplan
- set plan optgoal
- set [optCriteria]
- set plan opttimeoutlimit
- set plan replace
- set statistics simulate
- set metrics_capture
- set prefetch
- set parallel_degree number
- set process_limit_action
- set resource_granularity number
- set scan_parallel_degree number
- set repartition_degree number

CHAPTER 10

Supported Security Features

This chapter describes the security functionality of Adaptive Server Enterprise version 15.0.2.

Adaptive Server version 15.0.2 also includes new functionality in its Encrypted Columns feature. See Chapter 2, “Encrypted Columns,” for a detailed description.

In addition, information about security features released with Adaptive Server Version 12.5.4 is included in Adaptive Server Version 15.0.2. See Part 3, Chapter 25, “Security Enhancements.”

PAM support in 64-bit Adaptive Server on AIX

Adaptive Server version 15.0.2 on AIX 5.2 64-bit supports Pluggable Authentication Module-based User Authentication (PAMUA). Contact your support representative to make sure you have the latest patch for PAM available on your IBM host.

To enable PAMUA in 64-bit Adaptive Server 15.0.2 on AIX 5.2, you must supply the PAM module in */etc/pam.conf* file. For example:

```
ase auth required /usr/lib/security/pam_aix
```

Auditing enhancements

Adaptive Server version 15.0.2 introduces two additional auditing security features.

Hiding system stored procedure and command password parameters

When auditing is configured and enabled, and the sp_audit option 'cmdtext' is set, system stored procedure and command password parameters are replaced with a fixed length string of asterisks in the audit records contained in the audit logs.

For example, executing:

```
sp_password 'oldpassword', 'newpassword'
```

when auditing is enabled and sp_audit cmdtext is set, results in output similar to:

```
sp_password '******', '******'
```

This protects passwords from being seen by other with access to the audit log.

Monitoring failed login attempts

The new audit option login_locked and the event Locked Login (value 112) record when a login account is locked due to exceeding the configured number of failed login attempts. This event is enabled when audit option login_locked is set. To set login_locked, enter:

```
sp_audit "login_locked","all","all","ON"
```

If the audit tables are full and the event cannot be logged, a message with the information is sent to the errorlog.

The hostname and network IP address are included in the audit record. Monitoring the audit logs for the Locked Login event (112) helps to identify attacks on login accounts.

SSL support

Adaptive Server version 15.0.2 supports Secure Sockets Layer (SSL) on Windows 2003 X64 Enterprise Edition.Windows 2003 X64 implements SSL functionality using OpenSSL libraries.

SSL functionality is the same as Adaptive Servers on other platforms. It supports the same cipher suites and pre-defined lists as shown in Chapter 19, "Confidentiality of Data," in the *System Administration Guide*. OpenSSL libraries for Adaptive Server on Windows 2003 X64 communicates with SSLPlus clients and servers using the same cipher suite names as used in SSLPlus. The cipher suite names conform to the Transport Layer Security (TLS). TLS is an enhanced version of SSL 3.0, and is compatible with the SSL version 3.0 Cipher Suites.

Password security

Adaptive Server 15.0.2 adds a higher level of security to the existing password protection mechanisms through:

- Stronger encryption for passwords sent across the network
- Stronger encryption for passwords stored in syslogins (on disk) and in memory
- Managing login and password use with new time stamp data and additional account reporting

Securing login passwords on the network

Adaptive Server allows the use of asymmetric encryption to securely transmit passwords from client to server using the RSA public key encryption algorithm. Adaptive Server generates the asymmetric key pair and sends the public key to clients that use the new login protocol. The client encrypts the user's login password with the public key before sending it to the server. The server, decrypts the password with the private key to begin the authentication of the client connecting.

You can configure Adaptive Server to require clients to use this protocol. Set the Adaptive Server configuration parameter net password encryption reqd to require all username- and password-based authentication requests to use RSA asymmetric encryption.

Generating an asymmetric key pair

Adaptive Server generates a new key pair:

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

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

To generate the key pair on demand, use:

```
sp_passwordpolicy "regenerate keypair"
```

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

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

```
sp_passwordpolicy "regenerate keypair", “datetime string”
```

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

Configuring the encryption algorithm with *net password encryption reqd*

Adaptive Server can restrict login authentication to use only RSA encryption algorithm or the Sybase proprietary algorithm with configuration parameter net password encryption reqd. Table 10-1 describes valids values for net password encryption reqd.

Table 10-1: Values and descriptions for net password encryption reqd

Value	Description
0	Allows the client to choose the encryption algorithm used for login passwords on the network, including no password encryption. This is the default value for this configuration parameter and provides functionality most similar to earlier releases. This allows the choice of network password encryption to be established by the client application.
1	Restricts clients to use either RSA or Sybase proprietary encryption algorithms to encrypt login passwords on the network. This provides an incrementally restrictive setting that allows older clients to connect with the Sybase proprietary algorithm and new clients to connect with the stronger RSA algorithm. A client that attempts to connect without using password encryption will fail.
2	Restricts clients to use only the RSA encryption algorithms to encrypt login passwords on the network. This provides strong RSA encryption of passwords and requires use of newer clients. A client that attempts to connect without using the RSA encryption will fail.

The client receives the following error message when a connection is refused because network password encryption is required:

```
Msg 1640, Level 16, State 2:
Adaptive Server requires encryption of the login
password on the network.
```

Server option "net password encryption"

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

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

Adaptive Server uses either RSA or Sybase proprietary algorithms when this server option is set to “true.” The command to enable “net password encryption” is:

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

The setting is stored in master..sysservers and the value of server options are displayed using sp_helpserver stored procedure.

In this release, the default value for net password encryption is "true" for any new server added using sp_addserver. During upgrade, Adaptive Server sets net password encryption to "true" for sysservers entries with an ASEnterprise class value. No other server classes are modified. This is done to improve password security between two communicating Adaptive Servers.

Note The administrator may reset net password encryption to false if you encounter problems establishing a connection to a server. However, if the option is set to false, your password is transmitted in clear text on the network.

Backward compatibility

- Sybase recommends that you use the RSA algorithm to protect passwords on the network.
- To use the RSA algorithm, you must have Adaptive Server version 15.0.2 and new Connectivity SDK clients (version 15.0 ESD#7 and later.) Sybase provides the net password encryption reqd configuration parameter and the net password encryption server option to allow settings equivalent to pre-15.0.2 release and maintain backward compatibility with older clients and older servers.
- Older clients that do not support RSA algorithm can set the property to encrypt passwords using the Sybase proprietary algorithm, available since pre-12.0 releases. Adaptive Server then uses the Sybase proprietary algorithm.
- New clients that support both RSA and Sybase proprietary algorithms can set properties for both algorithms. When communicating with such clients, Adaptive Server 15.0.2 uses RSA encryption. A pre-15.0.2 Adaptive Server uses Sybase's proprietary algorithm.
- See client documentation for details for RSA encryption algorithm, referred to as Extended Password Encryption feature in the sections "What's new in ESD #7" and "Extended Password Encryption" in *New Features Open Server 15.0 and SDK 15.0 for Microsoft Windows, Linux, and UNIX*. These documents describe how to enable password encryption (Sybase proprietary algorithm) or extended password encryption (RSA algorithm) in the client.

Securing login passwords stored on disk and in memory

Login passwords used by Adaptive Server to authenticate client connections are stored securely on disk as SHA-256 hash digest. The SHA-256 algorithm is a one-way encryption algorithm. The digest it produces can not be decrypted, making its storage on disk secure. To authenticate the user connection, the SHA-256 algorithm is applied to the password sent by the client and the result compared with the value stored on disk.

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

To ease the transition to the new on-disk encryption algorithm when migrating from pre-15.0.2 releases, Adaptive Server now includes the password policy allow password downgrade. Upon upgrade from pre-15.0.2 releases, the policy has value "1" to indicate that passwords are stored in both the Sybase proprietary algorithm used in earlier releases and the new SHA-256 algorithm used in Adaptive Server 15.0.2.

As long as passwords are stored in both old and new forms, Adaptive Server can be downgraded to Adaptive Server 15.0 or 15.0.1 without resetting user passwords. When the policy allow password downgrade is set to 0, passwords will only be stored in new SHA-256 form, which is incompatible with older releases. When downgrading to previous releases, only passwords stored in SHA-256 are reset to random passwords and stored in the old form compatible with older releases. For more information, see "Backward compatibility" on page 154.

Due to the superior password security of the SHA-256 algorithm, Sybase recommends only using SHA-256 as soon the administrator is certain that downgrade to an earlier release will not be done. But consider the trade-offs when making this decision; should there be a need to downgrade to a pre-15.0.2 release, it requires administrator intervention to unlock user login passwords.

syslogins and *syssrvroles* system tables

The existing password columns in the syslogins and syssrvroles tables in the master database are upgraded from varbinary(30) to varbinary(128):

```
master..syslogins (
  ...
 password varbinary(128) null, /* encrypted password */
  ... )
```

```
master..syssrvroles (
  ...
 password varbinary(128) null, /* encrypted password */
  ... )
```

Behavior changes on upgraded *master* database

In an upgraded Adaptive Server 15.0.2 master database, the server maintains encrypted passwords in syslogins catalogs using both old and new algorithms in the password column.

Users can call sp_displaylogin to determine which "Login password encryption" a login uses.See the description and examples in "sp_displaylogin" on page 166.

On first authentication of a login after an upgrade:

- The user authenticates using the contents of the password column and the old algorithm.
- Adaptive Server then updates the password column with the old encryption algorithm followed by the new encryption algorithm.

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

Behavior changes on new *master* database

In a new Adaptive Server 15.0.2 master database, or an upgraded master database after "allow password downgrade" is set to 0, the server maintains encrypted passwords in syslogins catalog using only the new algorithm in password column. Only the SHA-256 algorithm is used to authenticate the connection requests and for storing the password on disk.

The administrator can distinguish a server that has been upgraded and maintaining passwords with both old and new algorithms from a server with a new master database that only uses the new algoritm by executing:

```
sp_passwordpolicy "list", "allow password downgrade"
```

See "Showing the current value of allow password downgrade" on page 161 for more information.

Using only the SHA-256 algorithm

You can downgrade password downgrade is allowed when user password updates are retained in both the old and new encoding. The old encoding is kept for use after downgrade to prior Adaptive Server release (for example, 15.0.1).

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

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

Before executing this command, administrators should examine login accounts with sp_displaylogin to determine if the login account has been used, and whether the password is stored in the new SHA-256 encoding. If the password is not encoded with the new encoding, the account is locked by this action, and the password is reset with a generated password. To use the account again, administrator intervention is required to unlock the account and give the user a newly generated password.

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

Adaptive Server takes the following actions when the password downgrade period ends:

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

After the sp_passwordpolicy procedure completes, the following behavior is expected:

- The authentication of logins is done using the new algorithm only.
- Only the new password on disk structure for the password column is used.

- Attempts to use the locked logins will fail authentication. In order to use the logins that were locked, the administrator must unlock the login with sp_locklogin and you must use the password generated by sp_passwordpolicy. Alternatively, the administrator may prefer to assign a new password instead of the generated password for locked login accounts.

Example 1

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

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

The value “SYB-PROP” from the line Login Password Encryption: SYB-PROP indicates that only the old, Sybase proprietary password encryption is used for this account. This login has not been used since upgrade to Adaptive Server version 15.0.2 and would be locked and password reset if the procedure “sp_passwordpolicy 'set', 'allow password downgrade', 0” is executed.

After the first login to the account after upgrading to Adaptive Server 15.0.2, the line would changed to show value “SYB-PROP, SHA-256” to show that both old and new encryption is used:

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

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

After the "sp_passwordpolicy 'set', 'allow password downgrade', 0" procedure is executed, only the new encryption is used, and the following line appears:

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

Login accounts that show this value are fully transitioned to use the new, stronger, on-disk encryption algorithm.

Example 2

This example executes "sp_passwordpolicy 'set', 'allow password downgrade', 0" when all accounts have transitioned to new algorithm

When all passwords have been changed to use the new algorithm, executing the command shows no accounts reset or locked.

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

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

Example 3

This example executes "sp_passwordpolicy 'set', 'allow password downgrade', 0" with accounts that have not transitioned to new algorithm

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

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

Old password encryption algorithm found for login name login1000, suid 3,
ver1 =5, ver2 = 0, resetting password to EcJxKmMvOrDsC4
Old password encryption algorithm found for login name login999, suid 4,
ver1 =5, ver2 = 0, resetting password to MdZcUaFpXkFtM1
Old password encryption algorithm found for login name login998, suid 5,
ver1 =5, ver2 = 0, resetting password to ZePiZdSeMqBdE6
Old password encryption algorithm found for login name login997, suid 6,
ver1 =5, ver2 = 0, resetting password to IfWpXvGlBgDgW7
Old password encryption algorithm found for login name login996, suid 7,
ver1 =5, ver2 = 0, resetting password to JhDjYnGcXwObI8
Old password encryption algorithm found for login name login995, suid 8,
ver1 =5, ver2 = 0, resetting password to QaXlRuJlCrFaE6
Old password encryption algorithm found for login name login994, suid 9,
ver1 =5, ver2 = 0, resetting password to HlHcZdRrYcKyB2
Old password encryption algorithm found for login name login993, suid 10,
ver1 =5, ver2 = 0, resetting password to UvMrXoVqKmZvU6
Old password encryption algorithm found for login name login992, suid 11,
ver1 =5, ver2 = 0, resetting password to IxIwZqHxEePbX5
Old password encryption algorithm found for login name login991, suid 12,
```

```
ver1 =5, ver2 = 0, resetting password to HxYrPyQbLzPmJ3
Old password encryption algorithm usage eliminated from 10 login accounts,
changes are committed.
(return status = 1)
```

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

Expiring passwords when *allow password downgrade* is set to 0

An administrator can choose to "expire" passwords in syslogins at the end of the password downgrade period in order to void copies of password in backups, transaction logs, and other persistent storage where they were kept with the old algorithm.

To configure login passwords to expire, use:

sp_passwordpolicy "expire login passwords"[, "[*loginame* | *wildcard*]"

To configure role passwords to expire, use:

sp_passwordpolicy "expire role passwords"[, "[*rolename* | *wildcard*]"

See "sp_passwordpolicy" on page 168 for additional details about expiring password.

Expiration of passwords can be tuned to expire passwords of logins and roles with passwords that have not been changed after a datetime specified.

To configure stale login passwords to expire, use:

sp_passwordpolicy "expire stale login passwords", "*datetime*"

To configure stale role passwords to expire, use:

sp_passwordpolicy "expire stale role passwords", "*datetime*"

By executing sp_passwordpolicy "expire stale login passwords" and setting the *datetime* parameter with the value when the password downgrade period ended, then all login passwords that have not changed since that time will have their passwords expired. The benefit of expiring passwords after the password downgrade period ends is to require the user to change to a different password thereby making the old password useless. Additionally, this is done without requiring additional actions by the administrator.

A password that is encrypted with the old algorithm during the password downgrade period may remain in old pages, transaction logs, and other storage in the less securely encrypted form. An old password that is different than the current password prevents successful exploits on the old encryption algorithm by making the old password useless.

A stronger approach for the administrator to take is to lock stale logins or roles; however this would require the administrator to reset the password manually for legitimate users to access their login account again.

Showing the current value of *allow password downgrade*

You can obtain the current value of attribute allow password downgrade by using the following procedure call:

```
sp_passwordpolicy list, "allow password downgrade"
```

The output is a result set containing the current value and a message indicating its meaning.

For an upgraded master database that is still maintaining passwords in the old and new encodings, the result is:

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

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

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

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

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

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

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

value    message
```

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

Character set considerations

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

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

sp_password *old_passwd*, *new_passwd*

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

To address this limitation with character sets, Adaptive Server converts the clear text password to canonical form before encryption so that it can be used across platforms, chip architectures, and character sets.

The steps to put in canoncial form for storage in syslogins:

1 Convert clear text password string to UTF-16.

2 Convert the UTF-16 string to network byte order.

3 Append a small buffer (the salt) with random bytes to the password.

4 Apply SHA-256 hash algorithm.

5 Store digest, salt, and version in the password column.

At authentication time, the steps are:

1 Convert clear text password string to UTF-16.

2 Convert the UTF-16 string to network byte order.

3 Append the salt from the password column in syslogins to the password.

4 Apply the hash algorithm.

5 Compare results with password column in syslogins, if they match then authentication is successful.

Use of the new SHA-256 algorithm with the above steps allow passwords to be transported across platforms, chip architectures, and character sets.

Last login and locking inactive accounts

Improved security for user accounts is provided by tracking the creation date, the last login time for an account, and the ability to determine which accounts are stale and may be locked due to inactivity. Additional information is kept to record the reason for an account to be locked and the identity of the user who locked the account.

New columns in *syslogins*

The following new columns are defined to support Last Login and Locking Inactive Accounts:

```
master..syslogins (
    ...,
    lastlogindate      datetime null,
    crdate             datetime null,
    locksuid           int null,
    lockreason         int null,
    lockdate           datetime null,
    ... )
```

These new columns are readable by public—the same permissions as most other columns in syslogins including pwdate, the date, the password last changed, and status, which includes bits to identify locked accounts and expired passwords. Only the password and audflags columns control select permission to system roles sso_role or sa_role.

Behavior changes

The new columns allow an account owner or administrator to know if an account is locked, when it was locked, who locked it, and the reason why it was locked.

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

At login time, if the "enable last login updates" password policy option is set to "1," the lastlogindate column is set to the current datetime and the previous value of the column is stored in the PSS of the login session.The update to syslogins and the PSS can occur at each login to Adaptive Server.The default value for this option in a new master database or an upgraded database is "1." The administrator can choose to disable this option by executing the procedure:

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

Global T-SQL variable *@@lastlogindate* is available to each user login session. A datetime datatype, its value is the lastlogindate column for the login account before the current session was established. This variable is specific to each login session and can be used by that session to determine the date and time of the previous login to the account. If the account has not been used previously or "enable last login updates" is 0, then the value of *@@lastlogindate* is NULL.

Updates to column lastlogindate in syslogins are not logged in the transaction log, because adding a log record for every login would cause master database to fill up quickly.

An administrator with sso_role can lock login accounts that have been inactive for a given number of days using the following sp_locklogin command:

sp_locklogin 'all', 'lock', [@except], '*number of inactive days*'

This command has no effect if "enable last login updates" is set to "0" or the value of the lastlogindate column is NULL. The value of *number of inactive days* can be in the range 1 to 32767 (days).

When a login is locked using such a command, the lockreason column is updated with the reason. The value of the lockdate column is set to the current datetime.

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

The lockdate, locksuid and lockreason columns are set internally by Adaptive Server. Table 10-2 describes the reasons and the value of locksuid in this release.

Table 10-2: The reasons and values of locksuid

Values for lockreason	Value for locksuid	Description of lockreason of account
NULL	NULL	Account has not been locked.
0	suid of caller of sp_locklogin	Account locked by locksuid by manually executing sp_locklogin.

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

Auditing

A new audit option “password” audits these actions.

```
sp_passwordpolicy 'set', 'option_name', 'option_value'
sp_passwordpolicy 'clear', 'option_name'
sp_passwordpolicy 'expire login passwords'
sp_passwordpolicy 'expire stale login passwords'
sp_passwordpolicy 'regenerate keypair'
sp_passwordpolicy 'expire role passwords'
sp_passwordpolicy 'expire stale role passwords'
```

Note The “list” option is not an audited action.

Configuration

Adaptive Server includes a new configuration parameter called “net password encryption reqd.” The administrator can set this new configuration parameter to require all incoming login authentication requests using Adaptive Server authentication to use the encryption described in this chapter.

The valid values for this parameter are described in Table 10-1 on page 153.

Changed and additional system stored procedures

sp_addlogin

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

sp_displaylogin

sp_displaylogin is enhanced to display the encryption version(s) used for a login. For example, when both old and new encryption is used during the password downgrade period, the output of sp_displaylogin has the new line "Password encryption."

In this example, both the old Sybase proprietary encryption algorithm is used as well as the new algorithm SHA-256:

```
1> sp_displaylogin 'mylogin'
2> go

Suid: 121
Loginame: mylogin
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
        sybase_ts_role (default ON)
Locked: NO
Date of Last Password Change: Aug 10 2006 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: NONE
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

When only the new encryption is used, the output of sp_displaylogin has the line "Login password encryption: SHA-256":

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
 ...
Authenticate with: NONE
Login password encryption: SHA-256
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

When a login has not occurred after upgrade, the old encryption is still in place, and the output of sp_displaylogin has the line "Login password encryption: SYB-PROP":

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
 ...
Authenticate with: NONE
Login password encryption: SYB-PROP
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

When a login has been locked, sp_displaylogin shows the date, reason, and login that locked the account.The lastlogindate value is also displayed:

```
1> sp_displaylogin 'mylogin'
2> go
```

```
Suid: 121
Loginame: mylogin
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
        sa_role (default ON)
        sso_role (default ON)
        oper_role (default ON)
        sybase_ts_role (default ON)
Locked: YES
        Date when locked: Aug 18 2006 9:15AM
        Reason: Account locked by Adaptive Server due
to failed login attempts reaching max failed logins.
        Locking suid: 121
Date of Last Password Change: Aug 10 2006 11:17AM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
```

```
Maximum failed logins: 3
Current failed login attempts: 3
Authenticate with: ANY
Login password encryption: SYB-PROP, SHA-256
Last login date : Aug 17 2006 5:55PM
(return status = 0)
```

sp_locklogin

The "lock" option to sp_locklogin, when used with a value for "*number of inactive days*," locks inactive accounts that have not authenticated within that period. The following example locks all login accounts that have not authenticated within the past 60 days:

```
sp_locklogin 'all', 'lock', NULL, 60
```

The privileged role sso_role is required to lock logins.

This command has no effect if the sp_passwordpolicy option "enable last login updates" is set to "0".

The value for "*number of inactive days*" can be 1 to 32767 days.

The new syslogins columns lockdate, locksuid and lockreason are updated at time of locking/unlocking a login.

sp_passwordpolicy

The following sp_passwordpolicy options are new to Adaptive Server 15.0.2. sp_passwordpolicy was introduced in Adaptive Server 12.5.4. All sp_passwordpolicy options from 12.5.4 are also included in Adaptive Server 15.0.2. For more information, see Chapter 25, "Security Enhancements."

- The following syntax expires the password of a role, all roles or roles matching a wild-card pattern. The column status in master database catalog syssrvroles is updated with a status bit ROLE_EXPIRED (0x4) to indicate the password is expired:

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

- The following syntax expires login passwords, all logins or logins matching a wild card pattern. The column status in master database catalog syslogins is updated with a status bit LOGIN_EXPIRED (0x4) to indicate the password is expired.

  ```
  sp_passwordpolicy "expire login passwords", "[login name | wildcard]"
  ```

- The following syntax expires login passwords have not been changed after a datetime specified. The column status in master database catalog syslogins is updated with a status bit LOGIN_EXPIRED (0x0004) to indicate that the password is expired. See “Entering Date and Time Data” in Adaptive Server 15.0 *Reference Manual: Building Blocks*, Chapter 1, “System and User Defined Datatypes” for an explanation of how datetime values are entered.

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

- The following syntax expires role passwords have not been changed after a datetime specified. The column status in master database catalog syssrvroles is updated with a status bit ROLE_EXPIRED (0x4) to indicate the password is expired.

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

- The following syntax generates the asymmetric key pairs to be used for network login password encryption.There is no catalog update for this option; the actions occur only in memory fields.

  ```
  sp_passwordpolicy "regenerate keypair"
  ```

- New password policy option “enable last login updates” is used to enable or disable code in Adaptive Server authentication that records the timestamp when each login occurs.The first parameter “set” sets the value of this attribute.The parameter “list” displays the current value of the attribute, and the parameter "clear" deletes the row from sysattributes. On upgrade or new installation, this attribute does not exist in sysattributes. The login timestamp occurs when the attribute row does not exist or has a value of 1. The login timestamp is not maintained if the attribute value is 0.

  ```
  sp_passwordpolicy "set", "enable last login updates", 1 or 0
  sp_passwordpolicy "list", "enable last login updates"
  sp_passwordpolicy "clear", "enable last login updates"
  ```

- The following syntax ends the password downgrade period. During the password downgrade period, passwords are stored in syslogins in both old and new encodings to allow user passwords to retained if the server is downgraded, for example, to Adaptive Server 15.0.2.

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

System table changes

This section describes changes to Adaptive Server system tables.

Changes to syslogins

Column password is readable only by a user with sso_role, and this behavior is unchanged in Adaptive Server 15.0.2.

```
master..syslogins (
 ...,
 password varbinary(128) null, /* encrypted password */
 lastlogindate datetime null,
 crdate datetime null,
 locksuid int null,
 lockreason int null,
 lockdate datetime null,
 ...)
```

New columns lastlogindate, crdate, locksuid, lockreason, lockdate do not have read access restrictions. See “New columns in syslogins” on page 163 for more information.

Changes to syssrvroles

Column password is readable only by a user with sso_role, this behavior is unchanged in Adaptive Server 15.0.2.

```
master..syssrvroles (
 ...
 password varbinary(128) null, /* encrypted password */
 ...)
```

Installation, upgrade and downgrade changes

The ability to upgrade to Adaptive Server 15.0.2 to use new password encryption while keeping the ability to downgrade to an earlier release is supported in Adaptive Server 15.0.2. This password downgrade period is introduced only for upgrade in the event that problems arise after upgrade and the administrator chooses to downgrade back to Adaptive Server 15.0.1, for example.The ability to downgrade passwords is enabled by default upon upgrade and remains in effect until an administrator chooses to end this by disabling it.While enabled, only the new functionality is lost by downgrade to Adaptive Server 15.0.1, the password changes made by users are not lost. All changes take place in master database; no user databases are affected.

Note If you run sp_downgrade then shutdown the server and for some reason reboot the 15.0.2 server all the downgrade changes are undone. In this case you must run sp_downgrade again.

Installation

The master database contains the syslogins and syssrvroles tables. When the new encryption algorithms and password policies are enabled on Adaptive Server, additional disk space in master database and transaction log is needed.The DBA should use the alter database command to add sufficient space to the master database and transaction log to handle the additional disk requirements for the user population of the site.

The space for syslogins in master database is estimated to increase by about 30% for the same number of users. The maximum row length is increased 135 bytes per login account (row in syslogins). The ratio of rows per page has decreased from about 16 rows per 2k page to 12 rows per 2k page between Adaptive Server versions 15.0.1 and 15.0.2. During the period when the value of “allow password downgrade” is 1, when both old and new password encryption algorithms are used, the ratio further decreases to about 10 rows per 2K page.

For example, if a server has 1,000 login accounts in Adaptive Server 15.0.1 and data pages fit into 59 pages, it may require about 19 additional pages in Adaptive Server 15.0.2 on a new master database or 33 pages additional pages if upgraded from 15.0.1 and the value of “allow password downgrade” is 1.

Additional space in transaction log is required for the updated password column at first login (829 2K pages per 1,000 logins), for password changes made by users and during upgrade and downgrade actions (about 343 pages per 1,000 logins). To ensure that sufficient log space is available, verify that there is about one page (2K page) of free log space per login available before password upgrade or downgrade actions are taken, and while users make their first login to Adaptive Server 15.0.2.

On bootstrap of a new master database, the value for "allow password downgrade" is NULL.

Note This value is different for a bootstrap of master databases than for an upgraded master database.

Upgrade

The following actions are taken during an upgrade from an earlier Adaptive Server release (not in a new master database):

- Schema changes to syslogins and syssrvroles.
- Add a new row to sysattributes for "allow password downgrade," set with value 1 (default). The ability to downgrade passwords is kept on upgraded master databases until the stored procedure "sp_passwordpolicy set, 'allow password downgrade', 0" is called to disable it. The system attribute "allow password downgrade" is enabled by default upon upgrade.
- Add a new row to sysattributes for "enable last login updates," set with value 1 (default).The update of syslogins column lastlogindate is controlled by this attribute value.The administrator can disable this behavior by calling stored procedure "sp_passwordpolicy set, 'enable last login updates', 0."

Downgrade

Downgrade to Adaptive Server release 15.0 or 15.0.1 is supported, and can happen without the administrator taking action for each login account.

If “allow password downgrade” is 0 or NULL, or a password has otherwise been stored in syslogins with only the new SHA-256 algorithm, then the password is reset and the account is locked during downgrade. This will require administrator action before the account may be used again. Use sp_displaylogin on login accounts to determine what algorithm is used or sp_downgrade "prepare" to determine what accounts will be reset.

To know what actions will be taken and to verify that sp_downgrade may run successfully, you can perform a dry run using the “prepare” option, such as:

```
1>  sp_downgrade 'prepare','15.0.1',1
2> go

Checking databases for downgrade readiness.

There are no errors which involve encrypted columns.

Allow password downgrade is set to 0. Login passwords
may be reset, if old encryption version of password is
not present.

Warning: New password encryption algorithm found for
login name user103, suid 103 .

Password will be reset during the downgrade phase.

sp_downgrade 'prepare' completed.
(return status = 0)
```

In the above example, the login “user103” was found to have only the new password format that is not used on previous releases. If downgrade occurs, the password will be reset to a random password and the account locked.

This procedure runs through actions and report readiness for downgrade to occur. The downgrade action does not occur if prepare does not succeed. For login passwords, it will report which passwords will be reset.

Before executing sp_downgrade, Sybase recommends removing the login 'probe' from syslogins. To remove the login, connect to Adaptive Server as a System Security Officer or System Administrator, and execute:

```
sp_droplogin 'probe'
go
```

If the login has user entries in databases, use the following command to drop users from databases and then drop the login.

```
use master
go
```

```
sp_dropuser  'probe'
go
```

The 'probe' login will be re-created when the *installmaster* script is run on the downgraded server.

Before executing sp_downgrade, Sybase recommends that you drop statistics for syslogins and syssrvroles. This avoids invalid column information, such as the length of password column, in sysstatistics after you perform a downgrade.

To drop statistics for syslogins and syssrvroles, enter:

```
1> delete statistics master..syslogins
2> delete statistics master..syssrvroles
3> go
```

The actions to downgrade password occur when stored procedure sp_downgrade is executed. For example:

```
1> sp_downgrade 'downgrade','15.0.1',1
2> go

Checking databases for downgrade readiness.
There are no errors which involve encrypted columns.

Allow password downgrade is set to 0. Login passwords may be reset, if old encryption
version of password is not present.
Warning: New password encryption algorithm found for login name user103, suid 103 .
Password will be reset during the downgrade phase.

Executing downgrade step 1 [sp_passwordpolicy 'downgrade'] for :
- Database: master (dbid: 1)

New password encryption algorithm found for login name user103, suid 103.
Resetting password to 'ZdSuFpNkBxAbW9'.

Total number of passwords reset during downgrade = 1

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

In the above example, the execution of sp_downgrade resulted in the locking and password reset of login user103. The random password generated by Adaptive Server is shown only to the client executing sp_downgrade. The administrator can redirect this output to a file so that these passwords are retained, or the administrator can reset them manually after rebooting on the downgraded master.

Additional messages appear in the errorlog to identify steps that occurred during sp_downgrade and any system errors that may occur. Example errorlog output for the example downgrade procedure follows:

```
00:00000:00006:2007/05/21 05:34:07.81 server  Preparing ASE downgrade from 1502 to 1501.
00:00000:00006:2007/05/21 05:35:59.09 server  Preparing ASE downgrade from 1502 to 1501.
00:00000:00006:2007/05/21 05:35:59.19 server  Starting downgrading ASE.
00:00000:00006:2007/05/21 05:35:59.20 server  Downgrade : Downgrading login passwords.
00:00000:00006:2007/05/21 05:35:59.22 server  Downgrade : Starting password downgrade.
00:00000:00006:2007/05/21 05:35:59.23 server  Downgrade : Removed sysattributes rows.
00:00000:00006:2007/05/21 05:35:59.23 server  Downgrade : Updated 1 passwords.
00:00000:00006:2007/05/21 05:35:59.24 server  Downgrade : Removed columns in syslogins -
lastlogindate, crdate, locksuid, lockreason, lockdate are removed.
00:00000:00006:2007/05/21 05:35:59.26 server  Downgrade : Truncated password lengths.
00:00000:00006:2007/05/21 05:35:59.28 server Downgrade : Successfully completed password
downgrade.
00:00000:00006:2007/05/21 05:35:59.28 server  Downgrade : Marking stored procedures to
be recreated from text.
00:00000:00006:2007/05/21 05:36:03.69 server  Downgrade : Dropping Sysoptions system
table.
00:00000:00006:2007/05/21 05:36:03.81 server  Downgrade : Setting master database minor
upgrade version.
00:00000:00006:2007/05/21 05:36:03.83 server  Downgrade : Setting user databases minor
upgrade version.
00:00000:00006:2007/05/21 05:36:03.90 server  ASE downgrade completed.
```

This procedure makes the catalog changes and modifies password data to allow reverting to Adaptive Server version 15.0.1. The server must be in single-user mode in order to successfully execute sp_downgrade. A dataserver started with "-m" command line option starts the server in single-user mode and allows only the SA to login.

After sp_downgrade is run, the only safe thing to do is shut down the server to avoid new logins or other actions that may modify data or system catalogs. If restarting Adaptive Server at version 15.0.2 after sp_downgrade has successfully executed and server is shutdown, internal upgrade actions are performed again and the changes to system catalogs are upgraded to version 15.0.2 level. If Adaptive Server 15.0.2 was restarted before booting the 15.0.x release to which you are reverting it is necessary to rerun sp_downgrade.

See details on other aspects of downgrade in the the "Downgrade" section of the Adaptive Server release bulletin.

High Availability considerations

The password security feature impacts configuration of High Availability (HA) and the behavior of passwords in syslogins between primary and companion servers.

HA configuration

The primary and companion servers must have equivalent "allow password downgrade" values before they are configured for HA. A new quorum attribute "allow password downgrade" has been added to check whether the value of "allow password downgrade" is same on both primary and secondary servers. This HA advisory check succeeds when the value for the quorum attribute is the same, and HA advisory check fails when the values are different.

For example if value of "allow password downgrade" on primary server is set to 1 and on secondary server it is set to 0, then you see an output of sp_companion such as:

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

Step: Access verified from Server:'secondary_server' to
Server:'primary_server'.
Step: Access verified from Server:'primary_server' to
Server:'secondary_server'.
Msg 18836, Level 16, State 1:
Server 'secondary_server', Procedure 'sp_companion', Line 392:
Configuration operation 'configure' can not proceed due to Quorum Advisory Check
failure. Please run 'do_advisory' command to find the incompatible attribute
and fix it.

Attribute Name         Attrib Type      Local Value   Remote Value   Advisory
--------------         -----------      -----------   ------------   --------
allow password downg   allow password             0              1          2

(1 row affected)
(return status = 1)
```

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

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

The administrator must execute the sp_passwordpolicy 'allow password downgrade' option independently on both the primary and secondary servers to get the value in sync and then configure using sp_companion, in order to ensure that the both the servers are in same state. At any given point of time the two servers should have same value set for "allow password downgrade" option in order to ensure proper behavior.

Changed Behavior with passwords in syslogins with HA

After upgrading to Adaptive Server 15.0.2 and successful configuring HA, on the first connection to the primary server, the password of the user login is updated on both the primary and companion servers. This synchronizes the login password on primary and companion with the same on-disk encryption format. This is done to avoid password reset or locking when the "allow password downgrade" period ends as described in sp_passwordpolicy and with password downgrade to earlier Adaptive Server 15.0 versions with sp_downgrade. By synchronizing the password encryption format, the login passwords can continue to be used without being reset or locked by sp_passwordpolicy or sp_downgrade.

After successfully setting up the HA environment, ending the "allow password downgrade" period must be done separately on the primary and companion servers. Similarly, if downgrade to a previous version of Adaptive Server 15.0 is needed, the sp_downgrade action must be executed separately on the primary and companion servers.

CHAPTER 11

Installing Monitoring Tables

Adaptive Server version 15.0.2 includes these enhancements to monitoring tables installation:

Installing monitoring tables on Adaptive Server 15.0.2

Note You no longer need to run the *installmontables* script to install the monitoring tables.

Versions of Adaptive Server earlier than 15.0.2 required you to run the *installmontables* script to install the monitoring tables. Adaptive Server version 15.0.2 includes the monitor tables installation in the *installmaster* script.

You must have the mon_role to access the monitor tables.

Remotely accessing and editing monitoring tables

Sybase provides *installmontables* as a sample script showing how to remotely access monitoring tables. Run *installmontables* to view the instructions for editing. For example:

```
isql -Usa -Psa_password -Sserver name -i $SYBASE/$SYBASE_ASE/scripts/
installmontables
---x---x-------------------------------------------------------------x---x---

It is no longer necessary to run this script to install the Monitoring
Tables.Monitoring Tables are now installed by the installmaster script.
```

```
This installmontables script is provided as a sample that can be copied and
modified to support remote access of Monitoring Tables. To do so you need to:

1) Replace all instances of @SERVER@ with the name of the remote ASE from which
monitoring data is to be obtained. Note that each remote ASE to be monitored
must be added to the local ASE's sysservers table using sp_addserver.

2) Create a database with the same name as the remote ASE. This database need
only be of the minimum size as these tables do not store any data.

3) Remove this header (i.e. these first 21 lines).

4) Run the script against the local ASE using the isql utility as follows:

   isql -Usa -P<password> -S<server name> -i<script name>

5) Retrieve remote monitoring data. E.g. to obtain monEngine information for an
ASE named REMASE you would execute the following SQL:

            use REMASE
            go
            select * from monEngine
            go
```

CHAPTER 12

Monitoring Tables for the Statement Cache

This chapter contains information about new monitor tables for the statement cache.

Overview

The Adaptive Server statement cache stores SQL text of ad-hoc update, delete and select statements and other statements likely to be reused. When the statement cache is enabled, these statements are converted into lightweight procedures and their plans are saved for reuse. When a new statement is issued, Adaptive Server searches the statement cache for a plan to reuse. If Adaptive Server finds a plan to reuse, it avoids recompiling the statement, leading to performance enhancements.

The introduction of literal parameterization in Adaptive Server version 15.0.1 allows it to recognize queries that are the same except for differences in literal values, saving recompiling costs while using statement cache. In addition to performance benefits, literal parameterization leads to enormous space reduction while storing the metrics and statements in the cache.

Adaptive Server version 15.0.2 introduces two new monitoring tables that allow you to easily analyze the contents of the statement cache. monStatementCache provides a summary snapshot of the statement cache. monCachedStatement shows detailed information about each statement cached.

The columns in each table allow two attributes, “counter” if the column has a counter value, and “reset” if the column can be reset using mechanisms like sp_sysmon.

The monStatementCache table

Table 12-1 provides information about the name, the SQL datatype, attributes and description for each column in monStatementCache. The columns in monStatementCache allow two attributes, counter if the column has a counter value, and reset if the column can be reset using mechanisms like sp_sysmon.

Table 12-1: monStatementCache

Name	Type	Attributes	Description
TotalSizeKB	int		The configured size of the statement cache in KB.
UsedSizeKB	int		The amount of the statement cache currently in use.
NumStatements	int		The number of statements in the statement cache.
NumSearches	int	Counter/Reset	The number of times the statement cache was looked up.
HitCount	int	Counter/Reset	The number of times the statement cache was looked up and a match was found.
NumInserts	int	Counter/Reset	The number of statements that were inserted into the statement cache.
NumRemovals	int	Counter/Reset	The number of times statements were removed from the statement cache. This value includes both statements that were removed via explicit purges or due to a replacement strategy.
NumRecompilesSchemaChanges	int	Counter/Reset	The number of recompiles due to schema changes in the tables referred to in the cached statements.
NumRecompilesPlanFlushes	int	Counter/Reset	Number of recompiles due to the plan flushes from the cache.

The monCachedStatement table

Table 12-2 provides information about the name, the SQL datatype, attributes and description for each column of monCachedStatement. The columns in monCachedStatement allow two attributes, “counter” if the column has a counter value, and “reset” if the column can be reset using mechanisms like sp_sysmon.

Use the contents of monCachedStatement to find detailed information about specific cached statements, including information about resources used during the previous executions of a statement, how frequently a statement is executed, the settings in effect for a particular plan, number of concurrent uses of a statement and so on. This information is helpful when troubleshooting, and when deciding which statements are to be retained in the cache.

Table 12-2: monCachedStatement

Name	Datatype	Attributes	Description
SSQLID	int		SSQLID contains a unique identifier for each cached statement. This value is treated as a primary key for monCachedStatement, and is used in the built-in functions. For more information about the built-in functions, see “Displaying the SQL plan for cached statements” on page 187. show_cached_plan and show_cached_text both use SSQLID to refer to individual statements in the cache.
HashKey	int		This column contains the hash value of the SQL text of the cached statement. A hash key is generated based on a statement’s text, and can be used as an approximate key for searching other catalogs.
UserID	int		Contains the user ID of the user who initiated the statement that has been cached. Cached plans cannot be re-used if the user ID does not match.
SUserID	int		Contains the server ID of the user who initiated the cached statement. Cached plans cannot be reused if the server ID does not match.
DBID	int		Contains the database ID of the database from which the statement was cached. Cached plans cannot be reused if the database ID does not match.

Name	Datatype	Attributes	Description
CachedDate	datetime		Contains the timestamp of the date and time when the statement was first cached.
LastUsedDate	datetime		Contains the timestamp of the date and time when the cached statement was last used. Use this information in conjunction with the CachedDate to determine how frequently this statement is used, and whether it is helpful to have it cached.
CurrentUsageCount	int	Counter	Displays the number of concurrent users of the cached statement.
StatementSize	int		Displays the size of the cached statement in bytes.
MaxUsageCount	int	Counter	Displays the maximum number of times the cached statement's text was simultaneously accessed.
SessionSettings			Along with the text, parameters, DBID, UserID and so on, the following session settings must also match when the cache is searched.
ParallelDegree	byte		
QuotedIdentifier	boolean		
TableCount	byte		
TransactionIsolationLevel	byte		
TransactionMode	byte		
SAAuthorization	boolean		
SystemCatalogUpdates	boolean		
ExecutionMetrics			Execution costs are incurred when the cached plan is used. These costs are measured in terms of the LIO(logical IO) and PIO(physical IO) values, execution, and elapsed times. This information can also be obtained from the captured query processing metrics, but this requires metrics capture to be explicitly enabled. In this table, the metrics are independently captured for the cached statements regardless of Adaptive Server's metrics capture settings.
MetricsCount	int		
MaxExecTime	int		
MinExecTime	int		

Name	Datatype	Attributes	Description
AvgExecTime	int		
MaxElapsedTime	int		
MinElapsedTime	int		
AvgElapsedTime	int		
MaxLIO	int		
MinLIO	int		
AvgLIO	int		
MaxPIO	int		
MinPIO	int		
AvgPIO	int		
NumRecompilesPlanFlushes	int	Counter	Contains the number of times the cached statement was recompiled because a plan was not found in the cache.
NumRecompilesSchemaChanges	int	Counter	Contains the number of times the statement was recompiled due to schema changes. Running update statistics on a table may result in changes to the best plan. This change is treated as a minor schema change. Recompiling a statement many times indicates that it is not effective to cache this particular statement, and that you may want to delete the statement from the statement cache to make space for some other, more stable, statement.
MaxPlanSize	int		Contains the size of the plan when it is in use, in KB.
MinPlanSize	int		Contains the size of the plan when it is not in use, in KB.
LastRecompiledDate	datetime		Contains the date when the statement was last recompiled, either due of schema changes or because the statement was not found in the statement cache.
UseCount	int		Contains the number of times the statement was accessed after it was cached.
HasAutoParams	boolean		This column contains either true or false. It is true if the statement has any parameterized literals.

Installing and configuring Adaptive Server for monitoring information collection

monStatementCache and monCachedStatement, along with all other monitoring tables, are installed automatically when you run the *installmaster* script as part of the Adaptive Server installation process. For more information about installing Adaptive Server and the *installmaster* script, see the *Installation Guide* for your platform.

Adaptive Server does not collect monitoring information by default,. You must first configure Adaptive Server to collect monitoring information before enabling statement cache monitoring.

To configure Adaptive Server to collect general monitoring information set the configuration parameter enable monitoring to 1. enable monitoring is a master switch that determines whether the other monitoring options are enabled. For more information about enable monitoring, see Chapter 5: Setting Configuration Parameters of the *System Administration Guide: Volume 1*.

You can configure Adaptive Server to collect the monitoring information on the statement cache using sp_configure 'enable stmt cache monitoring, 1'. When this option is enabled, the MDA tables monStatementCache and monCachedStatement display valid data.

Deleting statements from the statement cache

You can remove the statements in the statement cache using dbcc purgesqlcache. In previous versions of the Adaptive Server, dbcc purgesqlcache deleted the entire contents of the statement cache. Adaptive Server version 15.0.2 enhanced this command to to take in the statement ID as a parameter. The statement IDs are visible in the SSQLID column of monCachedStatement. When you specify the statement ID, only the corresponding statement is deleted from the cache.

The sytax for dbcc purgesqlcache is:

```
dbcc purgesqlcache (int SSQLID)
```

Displaying the SQL plan for cached statements

You can view the plan for a cached statement using these functions:

show_cached_plan (int *SSQLID*)

show_plan (int *option*, int *SSQLID*)

For a statement that is not performing well, you can change the plans by altering the optimizer settings or specifying an abstract plan.

When you specify the first int variable in the existing show_plan argument as "-1", show_plan treats the second parameter as a SSQLID.

Note It is possible for a single entry in the statement cache to be associated with multiple, and possibly different, SQL plans. show_plan displays only one of them.

Obtaining the hash key from the SQL text

A hash key is generated based on a statement's text, and acts as an approximate key for the search mechanism in the statement cache. Since other MDA tables display the statement's text with lengths exceeding MDA_SQLTEXTLEN in multiple rows, and monCachedStatement does not display the text of the SQL command, the hashkey can be used as an approximate key to look up these tables.

For information about viewing an the entire SQL text of a cached statement, see "Displaying text and parameter information for cached statements" on page 188.

Adaptive Server version 15.0.2 provides two functions to compute the hashkey effectively. Use the parse_text function to verify the validity of the SQL text before computing the hash key. The syntax is:

char* parse_text(char *sqltext, int prm_opt)

Valid values for the prm_opt option are:

- 1 – indicates that parse_text will auto-parameterize the output text.
- -1 – indicates that the current session settings for literal parameterization should be used to decide whether the input text is parameterized.

Use the HASH function to compute the hash key over the statement's text. For example:

```
select HASH('xor32', 'select * from syskeys')
```

Displaying text and parameter information for cached statements

You can view the statement text of a cached statement using the show_cached_text function. This function uses the statement ID as input and displays the text and parameter information of the corresponding statement.The syntax is:

show_cached_text(int SSQLID)

CHAPTER 13

Finding Slow Running Queries

Adaptive Server version 15.0.2 introduces ways to collect diagnostic information about slow running queries.

Adaptive Server includes the show_sqltext, tracefile, and export_options parameters for the set command parameters that enable you to collect diagnostic information about poorly-running queries without having to previously enable showplan or other investigatory parameters.

Saving diagnostics to a trace file

Once enabled, set tracefile saves all SQL text for the current session to the specified file, each SQL text batch appending to the previous batch.

The syntax to enable tracing is:

```
set tracefile file_name [off] [for spid
```

The syntax to disable tracing is:

```
set tracefile off [for spid]
```

Where:

- *file_name* – is the full path to the file in which you are saving the SQL text. If you do not specify a directory path, Adaptive Server creates the file in *$SYBASE*.

  Note If *file_name* contains special characters (":", "/", and so on) other than numbers and letters, you must include *file_name* in quotes. For example, this *file_name* must be in quotes because of the "/" for the directory structure:

  ```
  set tracefile '/tmp/mytracefile.txt' for 25
  ```

  If *file_name* does not contain special characters and you want to save it to *$SYBASE*, it does not require quotes. For example, this *file_name* does not need to be in quotes:

  ```
  set tracefile mytracefile.txt
  ```

- off – disables the tracing for this session or spid.
- *spid* – server process ID whose SQL text you want saved to a trace file. Only the users with the SA or SSO role can enable tracing for other spids. You cannot save the SQL text for system tasks (such as the housekeeper or the port manager).

Examples

- This example opens a trace file named *sql_text_file* for the the current session:

```
set tracefile '/var/sybase/REL1502/text_dir/sql_text_file'
```

  Subsequent outputs from set showplan, set statistics io, and dbcc traceon(100) are saved in *sql_text_file*.

- This example does not specify a directory path, so the trace file is saved in *$SYBASE/sql_text_file*:

  ```
  set tracefile 'sql_text_file' for 11
  ```

  Any SQL run on spid 11 is saved to this tracefile.

- This example saves the SQL text for spid 86:

  ```
  set tracefile
  '/var/sybase/REL1502/text_dir/sql_text_file' for 86
  ```

- This example disables set tracefile:

  ```
  set tracefile off
  ```

These are the restrictions for set tracefile:

- You cannot save the SQL text for system tasks (such as the housekeeper or the port manager).
- You must have the sa or sso roles, or be granted set tracing permission, to run enable or disable tracing.
- set tracefile is not allowed to open an existing file as a tracefile.
- During an SA or SSO session, if you enable set tracfile for a specific spid, all subsequent tracing commands executed take effect on that spid, not the SA or SSO spid.
- If Adaptive Server runs out of file space while writing the tracefile, it closes the file and disables the tracing.
- If an isql session starts tracing for a spid, but the isql session quits without disabling the tracing, another isql session can begin tracing this spid.
- Tracing occurs for the session for which it is enabled only, not for the session that enabled it.
- You cannot trace more than one session at a time from a single sa or sso session. If you attempt to open a tracefile for a session for which there is already a trace file open, Adaptive Server issues this error message: `tracefile is already open for this session.`
- You cannot trace the same session from multiple sa or sso sessions.
- The file storing the trace output is closed when the session being traced quits or when you disable tracing.
- Before you allocate resources for tracing, keep in mind that each tracing requires one file descriptor per engine.

Set options that save diagnostic information to a trace file

You can use set tracefile in combination with other set commands and options that provide diagnostic information for a better understanding of slow-running queries. These are the set commands and options that save diagnostic information to a file:

- set show_sqltext [on | off]
- set showplan [on | off]
- set statistics io [on | off]
- set statistics time [on | off]

- set statistics plancost [on | off]

These are the set options:

- set option show [normal | brief | long | on | off]
- set option show_lop [normal | brief | long | on | off]
- set option show_parallel [normal | brief | long | on | off]
- set option show_search_engine [normal | brief | long | on | off]
- set option show_counters [normal | brief | long | on | off]
- set option show_managers [normal | brief | long | on | off]
- set option show_histograms [normal | brief | long | on | off]
- set option show_abstract_plan [normal | brief | long | on | off]
- set option show_best_plan [normal | brief | long | on | off]
- set option show_code_gen [normal | brief | long | on | off]
- set option show_pio_costing [normal | brief | long | on | off]
- set option show_lio_costing [normal | brief | long | on | off]
- set option show_log_props [normal | brief | long | on | off]
- set option show_elimination [normal | brief | long | on | off]

Which sessions are being traced?

Use sp_helpapptrace to determine which sessions Adaptive Server is tracing. sp_helpapptrace returns the server process IDs (spids) for all the sessions Adaptive Server is tracing, the spids of the sessions tracing them, and the name of the tracefile.

The syntax for sp_helpapptrace is:

```
sp_helpapptrace
```

sp_helpapptrace returns these columns:

- traced_spid – spid of the session you are tracing.
- tracer_spid – spid of the session that traced_spid is tracing. Prints “exited” if the tracer_spid session has exited.
- trace_file – full path to the tracefile.

For example:

```
sp_helpapptrace
 traced_spid   tracer_spid         trace_file
--------------   -------------     ----------
11               exited             /tmp/myfile1
13               14                 /tpcc/sybase.15_0/myfile2
```

Rebinding a trace

If a session is tracing another session, but quits without disabling the tracing, Adaptive Server allows a new session to rebind with the earlier trace. This means that a sa or sso is not required to finish every trace they start, but can start a trace session, quit, and then rebind to this trace session

Displaying SQL text

set show_sqltext allows you to print the SQL text for ad-hoc queries, stored procedures, cursors, and dynamic prepared statements. You do not need to enable the set show_sqltext before you execute the query (as you do with commands like set showplan on) to collect diagnostic information for a SQL session. Instead you can enable it while the commands are running to help determine which query is performing poorly and diagnose their problems.

Before you enable show_sqltext, you must first enable dbcc traceon to display the output to standard out:

```
dbcc traceon(3604)
```

The syntax for set show_sqltext is:

```
set show_sqltext {on | off}
```

For example, this enables show_sqltext:

```
set show_sqltext on
```

Once set show_sqltext is enabled, Adaptive Server prints all SQL text to standard out for each command or system procedure you enter. Depending on the command or system procedure you run, this output can be extensive.

For example, if you run sp_who, Adaptive Server prints all SQL text associated with this system procedure (the output is abbreviated for space purposes):

```
sp_who
2007/02/23 02:18:25.77
SQL Text: sp_who
Sproc: sp_who, Line: 0
Sproc: sp_who, Line: 20
Sproc: sp_who, Line: 22
Sproc: sp_who, Line: 25
Sproc: sp_who, Line: 27
Sproc: sp_who, Line: 30
Sproc: sp_who, Line: 55
Sproc: sp_who, Line: 64
Sproc: sp_autoformat, Line: 0
Sproc: sp_autoformat, Line: 165
Sproc: sp_autoformat, Line: 167
Sproc: sp_autoformat, Line: 177
Sproc: sp_autoformat, Line: 188
. . .
Sproc: sp_autoformat, Line: 326
Sproc: sp_autoformat, Line: 332
SQL Text: INSERT
#colinfo_af(colid,colname,usertype,type,typename,collength,maxlength,autoform
at,selected,selectorder,asname,mbyte) SELECT
c.colid,c.name,t.usertype,t.type,t.name,case when c.length < 80 then 80 else
c.length end,0,0,0,0,c.name,0 FROM tempdb.dbo.syscolumns c,tempdb.dbo.systypes
t WHERE c.id=1949946031 AND c.usertype=t.usertype
Sproc: sp_autoformat, Line: 333
Sproc: sp_autoformat, Line: 334
. . .
Sproc: sp_autoformat, Line: 535
Sproc: sp_autoformat, Line: 0
Sproc: sp_autoformat, Line: 393
Sproc: sp_autoformat, Line: 395
. . .
Sproc: sp_autoformat, Line: 686
Sproc: sp_autoformat, Line: 688
SQL Text: UPDATE #colinfo_af SET maxlength=(SELECT
isnull(max(isnull(char_length(convert(varchar(80),fid)),4)),1) FROM
#who1result ), autoformat = 1, mbyte=case when usertype in (24, 25, 34, 35) then
1 else 0 end WHERE colname='fid'
Sproc: sp_autoformat, Line: 689
Sproc: sp_autoformat, Line: 690
. . .
Sproc: sp_autoformat, Line: 815
Sproc: sp_autoformat, Line: 818
SQL Text: SELECT
fid=right(space(80)+isnull(convert(varchar(80),fid),'NULL'),3),
```

```
spid=right(space(80)+isnull(convert(varchar(80),spid),'NULL'),4),
status=SUBSTRING(convert(varchar(80),status),1,8),
loginame=SUBSTRING(convert(varchar(80),loginame),1,8),
origname=SUBSTRING(convert(varchar(80),origname),1,8),
hostname=SUBSTRING(convert(varchar(80),hostname),1,8),
blk_spid=right(space(80)+isnull(convert(varchar(80),blk_spid),'NULL'),8),
dbname=SUBSTRING(convert(varchar(80),dbname),1,6),
tempdbname=SUBSTRING(convert(varchar(80),tempdbname),1,10),
cmd=SUBSTRING(convert(varchar(80),cmd),1,17),
block_xloid=right(space(80)+isnull(convert(varchar(80),block_xloid),'NULL'),1
1) FROM #who1result  order by fid, spid, dbname



Sproc: sp_autoformat, Line: 819
Sproc: sp_autoformat, Line: 820
Sproc: sp_autoformat, Line: 826
Sproc: sp_who, Line: 68
Sproc: sp_who, Line: 70
fid  spid  status   loginame     origname    hostname  blk_spid  dbname
tempdbnamecmd      block_xloid
---  ----  --------  --------   --------   --------  --------    -------------
------------       -----------
0    2     sleeping  NULL       NULL       NULL       0        master tempdb
DEADLOCK TUNE      0
0    3     sleeping  NULL       NULL       NULL       0        master tempdb
ASTC HANDLER       0
0    4     sleeping  NULL       NULL       NULL       0        master tempdb
CHECKPOINT SLEEP   0
0    5     sleeping  NULL       NULL       NULL       0        master tempdb
HK WASH            0
0    6     sleeping  NULL       NULL       NULL       0        master tempdb
HK GC           0
0    7     sleeping  NULL       NULL       NULL       0        master tempdb
HK CHORES          0
0    8     sleeping  NULL       NULL       NULL       0        master tempdb
PORT MANAGER       0
0    9     sleeping  NULL       NULL       NULL       0        master tempdb
NETWORK HANDLER    0
0    10    sleeping  NULL       NULL       NULL       0        master tempdb
LICENSE HEARTBEAT  0
0    1     running   sa         sa         echo       0        master tempdb
INSERT             0

(10 rows affected)
(return status = 0)
```

To disable show_sqltext, enter:

```
set show_sqltext off
```

Restrictions for show_sqltext

- You must have the sa or sso roles to run show_sqltext.
- You cannot use show_sqltext to print the SQL text for triggers.
- You cannot use show_sqltext to show a binding variable or a view name.

Retaining session settings

Adaptive Server's default behavior is to reset any set parameter changes that are set by a trigger or system procedure after they finish running. Enabling set export_options allows you to retain the session settings that are set by a system procedure or trigger for the duration of the session. The syntax for set export_options is:

set export_options [on | off]

For example, this enables set export_options:

```
set export_options on
```

This disables set export_options and returns Adaptive Server to the default behavior:

```
set export_options off
```

CHAPTER 14

Row-level locking for system tables

Adaptive Server version 15.0.2 introduces a new way to lock system tables.

Overview

Versions of Adaptive Server earlier than 15.0.2 used exclusive table locks on system tables while executing data definition language (DDL) and utility commands. The set of system tables Adaptive Server locked depended on the type of DDL operation you executed. If another DDL running concurrently tried to take a conflicting exclusive table lock on the same system table, this DDL had to wait to acquire the lock on any system catalogs. These DDL operations were executed serially.

This methodology impeded performance in temporary databases, where their DDL activity is very high also, and consquently their catalog contention is very high. This limited the Adaptive Server throughput for applications using temporary tables.

Adaptive Server version 15.0.2 uses row-level locking to resolve these issues:

- System-table contention, caused a bottleneck for many DDLs and utilities.
- tempdb contention. Because the system tables are locked at the row level, Adaptive Server 15.0.2 eliminates tempdb contention.

- Shared or exclusive table-level locks while executing DDLs and utilities. Earlier versions converted most system tables to data-only locking (DOL), but still created shared or exclusive table-level locks while executing DDLs and utilities. Using row-level locks for system tables eliminates this contention.

  Adaptive Server sets intent locks on catalogs only, which removes potential contention (An intent lock indicates that page-level or row-level locks are currently held on a table.).

- DDLs and utilities blocking each other. Adaptive Server 15.0.2 allows DDLs and utilities to run in parallel.

Earlier versions of Adaptive Server used table locks to achive system catalog synchronization. Adaptive Server 15.0.2 uses intent locks for table-level synchronization and row locks for row-level synchronization. Earlier releases of Adaptive Server locked the entire system catalog while performing operations on the object, so a single lock request was made. However, Adaptive Server version 15.0.2 requests locks for all applicable rows while performing operations on the object if there are multiple rows corresponding to an object in a system catalog.

This change means that Adaptive Server 15.0.2 requests more locks to perform the same operation than earlier releases, and increases the number of lock resources the system needs. Consequently, you may need to change the number of locks configuration option after you upgrade Adaptive Server.

Commands that take intent locks

These commands take intent locks in Adaptive Server 15.0.2 when they update a system table:

- create table
- drop table
- create index
- drop index
- create view
- drop view
- create procedure

- drop procedure
- create trigger
- drop trigger
- create default
- drop default
- create rule
- drop rule
- create function
- drop function
- create functional index
- drop functional index
- create computed column
- drop computed column
- select into
- alter table (all versions)
- create schema
- reorg rebuild

If two or more of these commands simultaneously access or update the same system table, their intent locks do not conflict with each other so they do not block on the system table.

The sp_fixindex and sp_spaceusage system procedures have also been updated to provide more information about the row-locked catalogs. See Chapter 20, "Changes to System Procedures, Functions, and Commands."

Updates to monitoring tables

Adaptive Server 15.0.2 adds information to the following monitoring tables.

monOpenPartition Activity

monOpenPartitionActivity reports information about the events for each partition in the housekeeper garbage collection queues. The columns added are:

- HkgcRequests – reports the total number of events queued for the partition.

A large value reported by HkgcRequests implies the sytem is generating lot of garbage for that particular partition or object.

- HkgcPending – reports the number of pending events for the partition.

  Large value reported by HkgcPending implies a lot of garbage is yet to be collected, although the housekeeper will clean the garbage up. However, if Adaptive Server is rebooted, all the entries in the housekeeper queues are lost and the garbage from those page is not collected when you next start Adaptive Server.

- HkgcOverflow – reports the number of partition events that overflowed

  A large value reported by HkgcOverflow implies the housekeeper queues are getting full and the garbage that is generated will not be cleaned because the housekeep can not schedule the job.

These columns report on per-partition basis, and give an idea of the amount of garbage generated in each partition for different objects.

monOpenObect Activity

monOpenObjectActivity reports information about the events for each object in the housekeeper garbage collection queues. The columns added are:

- HkgcRequests – reports the total number of events queued for the object.

  A large value reported by HkgcRequests implies the sytem is generating lot of garbage for that particular partition or object.

- HkgcPending – reports the number of pending events for the object.

  Large value reported by HkgcPending implies a lot of garbage is yet to be collected, although the housekeeper will clean the garbage up. However, if Adaptive Server is rebooted, all the entries in the housekeeper queues are lost and the garbage from those page is not collected when you next start Adaptive Server.

- HkgcOverflow – reports the number of object events that overflowed

  A large value reported by HkgcOverflow implies the housekeeper queues are getting full and the garbage that is generated will not be cleaned because the housekeep can not schedule the job.

monDeadLock

monDeadLock reports the locking information (filename and line number) collected from the lock record deadlock chain. The columns added are:

- HeldSrcCodeID – filename and line number where Adaptive Server requests the holding lock
- WaitSrcCodeID – filename and line where Adaptive Server requests the waiting lock.

monLocks

monLocks adds the SourceCodeID column, which reports locking information recorded in the lock record structure:

- SourceCodeID – Filename and line at which the lock is requested

CHAPTER 15 # *xmltable()*

This chapter describes the xmltable() function.

Overview

xmltable() extracts a sequence of multivalued elements from an XML document, and assembles a SQL table of those elements. A single call to xmltable() replaces a Transact-SQL loop that performs multiple calls to xmlextract() on each iteration. xmltable() is invoked as a derived table (a parenthesized subquery specified in the from clause of a different SQL query). Calling xmltable() is equivalent to executing a single xmlextract() expression for each row of the table generated by xmltable().

xmltable() is a generalization of xmlextract(). Both functions return data extracted from an XML document that is an argument in the function. The differences are:

- xmlextract() returns the data identified by a single XPath query.
- xmltable() extracts the sequence, or row pattern, of the data identified by an XPath query, and extracts from each element of that sequence the data identified by a list of other XPath queries, the column patterns. It returns all the data in a SQL table.

xmltable() and derived table syntax

This section shows the basic syntax of xmltable() and where and how to use xmltable().

xmltable()

Description

Extracts data from an XML document and returns it as a SQL table.

Syntax

xmltable_expression ::= xmltable
(*row_pattern* passing *xml_argument*
columns *column_definitions*
options_parameter)
row_pattern ::= *character_string_literal*
xml_argument ::= xml_expression | column_reference | variable_reference
column_definitions ::=
column_definition [{ , *column_definition* }]
column_definition ::=
ordinality_column | *regular_column*
ordinality_column ::= *column_name* datatype for ordinality
regular_column ::=
column_name datatype [default *literal*] [null | not null]
[path *column_pattern*]
column_pattern ::= *character_string_literal*
options_parameter ::=[,] option *option_string*
options_string ::= *basic_string_expression*

Derived table syntax Returns a SQL table from within a SQL from clause.

from_clause ::= from *table_reference* [, *table_reference*]...
table_reference ::= *table_view_name* | *ANSI_join* |*derived_table*
table_view_name::=See the select command in Reference Manual Volume 2, "Commands."
ANSI_join::=See the select command in Reference Manual Volume 2, "Commands."
derived_table ::=
(subquery) as *table_name* [(*column_name* [, *column_name*]...)|
xmltable_expression as *table_name*

Parameters

xml_argument

An expression, column reference, or variable, referring to an XML document.

for

A reserved XML keyword.

ordinality

A non-reserved XML keyword.

passing

A non-reserved XML keyword

row_pattern

An XPath query expression whose result is a sequence of elements from the specified document. The xmltable() call returns a table with one row for each element in the sequence.

columns
A non-reserved XML keyword.

column_name
The user-specified name of the column.

column_pattern
An XPath query expression that applies to an element of the sequence returned by the *row_pattern*, to extract the data for a column of the result table. If the *column_pattern* is omitted, the *column_pattern* defaults to the *column_name*.

ordinality_column
A column of datatypes integer, smallint, tinyint, decimal, or numeric, which indicates ordering of the elements in the input XML document.

regular_column
Any column that is not an ordinality column.

derived_table
A parenthesized subquery specified in the from clause of a SQL query.

path
A reserved XML keyword.

option
An *option_string*, defined in *XML Services*, and a reserved XML keyword.

Examples

Example 1 This example shows a simple xmltable() call with the document specified as a character-string literal:

```
select * from xmltable('/doc/item'
   passing '<doc><item><id>1</id><name>Box</name></item>'
      +'<item><id>2</id><name>Jar</name></item></doc>'
   columns id int path 'id',name varchar(20) path 'name') as items_table
id               name
-----            -----
1                Box
2                Jar

(2 rows affected)
```

Example 2 This example stores the document in a Transact-SQL variable, and references that variable in the xmltable() call:

```
declare @doc varchar(16384)
set @doc='<doc><item><id>1</id><name>Box</name></item>'
       +'<item><id>2</id><name>Jar</name></item></doc>'
```

```
select * from xmltable('/doc/item' passing @doc
   columns id int path 'id', name varchar(20) path 'name') as items_table

id            name
----          -----
1             Box
2             Jar

(2 rows affected)
```

Example 3 This example stores the document in a table and references it with a subquery:

```
select 100 as doc_id,
'<doc><item><id>1</id><name>Box</name></item><item><id>2</id>
   <name>Jar</name></item></doc>' as doc
into #sample_docs
select * from xmltable('/doc/item'
   passing(select doc from #sample_docs where doc_id=100)
   columns id int path 'id',name varchar(20) path 'name') as items_table

id            name
-------       -----
1             Box
2             Jar

(2 rows affected)
```

Example 4 If a row pattern returns an empty sequence, the result is an empty table:

```
select * from xmltable ('/doc/item_entry'
   passing '<doc><item><id>1</id><name>Box</name></item>'
      +'<item><id>2</id><name>Jar</name></item></doc>'
   columns id int path 'id',
      name varchar(20) path 'name') as items_table

id              name
-------         ------


(0 rows affected)
```

Example 5 The arguments following the columns keyword comprise the list of column definitions. Each column definition specifies a column name and datatype, as in create table, and a path, called the column pattern.

When the data for a column is contained in an XML attribute, specify the column pattern using "@" to reference an attribute. For example:

```
select * from xmltable ('/doc/item'
    passing '<doc><item id="1"><name>Box</name></item>'
        +'<item id="2"><name>Jar</name></item></doc>'
    columns id int path '@id', name varchar(20)) as items_table

id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

Example 6 A *column-pattern* is commonly the same as the specified *column_name*, for example name. In this case, omitting the column-pattern results in defaulting to the *column_name*:

```
select * from xmltable ('/doc/item'
    passing '<doc><item><id>1</id><name>Box</name></item>'
        +'<item><id>2</id><name>Jar</name></item></doc>'
    columns id int, name varchar(20)) as items_table

id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

Example 7 If you want a column pattern to default to the column name, in a column whose value is in an XML attribute, use a quoted identifier. You must then quote such identifiers when you reference them in the results:

```
set quoted_identifier on
select "@id", name from xmltable ('/doc/item'
    passing '<doc><item id="1"><name>Box</name></item>'
        +'<item id="2"><name>Jar</name></item></doc>'
    columns "@id" int, name varchar(20)) as items_table

@id              name
--------------------
1               Box
2               Jar

(2 rows affected)
```

Example 8 You can also use quoted identifiers to specify column names as default column patterns, using column names that are more complex XPath expressions. For example:

```
set quoted_identifier on
select "@id", "name/short", "name/full" from xmltable ('/doc/item'
    passing '<doc><item id="1"><name><short>Box</short>
            <full>Box, packing, moisture resistant, plain</full>
            </name></item>'
    +'<item id="2"><name><short>Jar</short>
            <full>Jar, lidded, heavy duty</full>
            </name></item></doc>'
    columns "@id" int, "name/short" varchar(20), "name/full" varchar(50))
    as items_table

@id              name/short             name/full
--------------------
1               Box                     Box, packing, moisture resistant,
plain
2               Jar                      Jar, lidded, heavy duty

(2 rows affected)
```

Example 9 The function text() is implicit in column patterns. This example does not specify text() in the column pattern for either the id or name column:

```
select * from xmltable ('/doc/item'
passing '<doc><item><id>1</id><name>Box</name></item>'
+ '<item><id>2</id><name>Jar</name></item></doc>'
columns id int path 'id', name varchar (20) path 'name') as items_table
id    name
---   --------
1     Box
2     Jar
(2 rows affected)
```

Example 10 Applying an implicit SQL converst statement to the data extracted from the column pattern, derives column values in datatype conversions.

```
select * from xmltable ('/emps/emp'
    passing '<emps>
<emp><id>1</id><salary>123.45</salary><hired>1/2/2003</hired></emp>'
+'<emp><id>2</id><salary>234.56</salary><hired>2/3/2004</hired></emp>'
       +'</emps>'
    columns id int path 'id', salary dec(5,2), hired date)
as items_table

id          salary        hired
```

```
-------     ---------     --------
1           123.45        Jan 2, 2003
2           234.56        Feb 3, 2004
(2 rows affected)
```

Example 11 You can use an *ordinality_column* in xmltable() to record the ordering of elements in the input XML document:

```
declare @doc varchar(16384)
set @doc = '<doc><item><id>25</id><name>Box</name></item>'
    +'<item><id>15</id><name>Jar</name></item></doc>'
select * from xmltable('/doc/item' passing @doc
    columns item_order int for ordinality,
       id int path 'id',
       name varchar(20) path 'name') as items_table
order by item_order

item_order          id             name
----------          ---            ----
1                   25             Box
2                   15             Jar
(2 rows affected)
```

Without the for ordinality clause and the item_order column, there is nothing in the id and name columns that indicates that the row of id 25 precedes the row of id 15. The for ordinality clause orders the output SQL rows the same as the ordering of the elements in the input XML document.

The datatype of an ordinality column can be any fixed numeric datatype: int, tinyint, bigint, numeric, or decimal. numeric and decimal must have a scale of 0. An ordinality column cannot be real or float.

Example 12 This example omits the <*name*> element from the second <*item*>. The name column allows names to be NULL by default.

```
select * from xmltable ('/doc/item'
    passing '<doc><item><id>1</id><name>Box</name></item>'
    +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) path 'name')
as items_table
id            name
------------------
1             Box
2             NULL

(2 rows affected)
```

Example 13 This example omits the <*name*> element from the second <*item*>, and specifies not null for the name column:

```
select * from xmltable ('/doc/item'
   passing '<doc><item><id>1</id><name>Box</name></item>'
   +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) not null path 'name')
as items_table

id              name
-----           --------
1               Box
Msg 14847, Level 16, State 1:
Line 1:
XMLTABLE column 0, does not allow null values.
```

Example 14 This example adds a default clause to the name column, and omits the <*name*> elements from the second <*item*>.

```
select * from xmltable ('/doc/item'
   passing'<doc><item><id>1</id><name>Box</name></item>'
   +'<item><id>2</id></item></doc>'
columns id int path 'id', name varchar(20) default '***' path 'name')
as items_table
id   name
---  ------
1    Box
2    '***'
(2 rows affected)
```

Example 15 This example shows SQL commands in which you can use an xmltable() call in a derived table expression.

This example uses xmltable() in a simple select statement:

```
select * from xmltable ('/doc/item'
   passing '<doc><item><id>1</id><name>Box</name></item>'
   +'<item><id>2</id><name>Jar</name></item></doc>'
columns id int path 'id',
   name varchar(20) path 'name') as items_table

id            name
--            ----
1             Box
2             Jar
(2 rows affected)
```

Example 16 This example uses xmltable() in a view definition. It stores a document in a table and references that stored document in a create view statement, using xmltable() to extract data from the table:

```
select 100 as doc_id,
'<doc><item><id>1</id><name>Box</name></item>'
   +'<item><id>2</id><name>Jar</name></item></doc>' as doc
into sample_docs
create view items_table as
   select * from xmltable('/doc/item'
      passing (select doc from sample_docs where doc_id=100)
      columns id int path 'id',
      name varchar(20) path 'name') as xml_extract
select * from items_table

id            name
------------------
1             Box
2             Jar
(2 rows affected)
```

Example 17 This example uses xmltable() in a cursor:

```
declare C cursor for
select * from xmltable ('/doc/item'
   passing (select doc from sample_docs where id=100)
   columns id int path 'id',
   name varchar(20) path 'name') as items_table
   go
declare @idvar int
declare @namevar varchar(20)
open C
while @@sqlstatus=0
begin
fetch C into @idvar, @namevar
print 'ID "%1!" NAME"%2!"', @idvar, @namevar
end
-------------------------
ID "1" NAME "Box"
ID "2" NAME "Jar"

(2 rows affected)
```

In applications that require multiple actions for each generated row, such as executing update, insert, or delete from other tables, you can process an xmltable() result with a cursor loop. Alternatively, store the xmltable() result in a temporary table and process that table with a cursor loop.

Example 18 This example uses xmltable() in select into:

```
select * into #extracted_table
from xmltable('/doc/item'
    passing (select doc from sample_docs where doc_id=100
    columns id int path 'id',
       name varchar(20) path 'name') as items_table

select * from #extracted_table

id            name
----          ------------
1             Box
2             Jar
```

Example 19 This example uses xmltable() in an insert command:

```
create table #extracted_data (idcol int, namecol varchar(20))
insert into #extracted_data
select * from xmltable('/doc/item'
    passing (select doc from sample_docs where doc_id=100)
    columns id int path 'id', name varchar(20) path 'name') as items_table
select * from #extracted_data

idcol          namecol
-----          ---------
1              Box
2              Jar
(2 rows affected)
```

Example 20 This example uses xmltable() in a subquery. xmltable() returns a SQL table, so the subquery must perform either an aggregation or a selection to return a single row and column for the subquery result.

```
declare @idvar int
set @idvar = 2
select @idvar,
(select name from xmltable ('/doc/item'
    passing(select doc from sample_docs where doc_id=100)
    columns id int path 'id',name varchar(20) path 'name') as item_table
where items_table.id=@idvar)
---------------------------
2                     Jar
(1 rows affected)
```

Example 21 This example joins an xmltable() result with other tables, using either multiple table joins in the from clause, or outer joins:

```
create table prices (id int, price decimal (5,2))
```

```
insert into prices values(1,123.45)
insert into prices values (2,234.56)
select prices.id,extracted_table.name, prices.price
from prices,(select * from xmltable('/doc/item'
   passing (select doc from sample_docs where doc_id=100)
   columns id int path 'id', name varchar(20) path 'name') as a) as
extracted_table
where prices.id=extracted_table.id
id name  price
-- ----  -----
1  Box   123.45
2  Jar   234.56
(2 rows affected)
```

Example 22 This example uses xmltable(), with a lateral reference to a column existing in a preceding table in the same from clause as xmltable():

```
create table deptab (col1 int, col2 image)
insert deptab values (1, '<dept>
                          <dept-id>1</dept-id>
                          <dept-name>Finance</dept-name>
                          <employees>
<emp><name>John</name><id>e11</id></emp>
<emp><name>Bela</name><id>e12</id></emp>
<emp><name>James</name><id>e13</id></emp></employees></dept>')

insert deptab values (2, '<dept>
                        <dept-id>2</dept-id>
                          <dept-name>Engineering</dept-name>
                          <employees>
<emp><name>Tom</name><id>e21</id></emp>
<emp><name>Jeff</name><id>e22</id></emp>
<emp><name>Mary</name><id>e23</id></emp></employees></dept>')

select id, empname from deptab, xmltable ('/dept/employees/emp' passing
deptab.co12 columns empname varchar (8) path 'name', id varchar (8)
path 'id') as sample_tab

id                    empname
---------             -----------
e11                   John
e12                   Bela
e13                   James
e21                   Tom
e22                   Jeff
e23                   Mary
(6 rows affected)
```

Usage

- xmltable() is a built-in, table-valued function.
- The syntax of derived tables requires you to specify a table name, even if you do not reference it. Therefore, each xmltable() expression must also specify a table name.
- The argument following passing is the input XML document.
- The result type of an xmltable() expression is a SQL table, whose column names and their datatypes are specified by *column_definitions.*
- To process documents, you can apply xmltable() to the XML document in each row of a table of XML documents.
- These keywords are associated with xmltable():
  - Reserved: for, option, xmltable, path
  - Not reserved: columns, ordinality, passing
- The expressions in the arguments of an xmltable() call can reference the column names of preceding tables in the from clause containing the xmltable() call. Only tables that precede the xmltable() call can be referenced. Such a reference, to a column of a preceding table in the same from clause, is called a *lateral reference*. For example:

```
select * from T1, xmltable(...passing T1.C1...)
       as XT2, xmltable(...passing XT2.C2...)as XT3
```

  The reference to T1.C1 in the first xmltable() call is a lateral reference to column C1 of table T1. The reference to XT2.C2 in the second xmltable() call is a lateral reference to column C2 of the table generated by the first xmltable() call.
- You cannot use xmltable() in the from clause of an update or delete statement. For example, the following statement fails:

```
update T set T.C=...
from xmltable(...) as T
where...
```

- Datatypes in *regular_columns* can be of any SQL datatype.
- To handle XML data whose format is not suitable for a SQL convert function, extract the data to a string column (varchar, text, image, java.lang.String).
- The extracted XML data for the column must be convertible to the column datatype, or an exception is raised.

- If a column pattern returns an empty result, the action taken depends on the `default` and `{null | not null}` clauses.
- The literal following a default in a *regular_column* must be assignable to the datatype of the column.
- There can be no more than one *ordinality_column*; the datatype specified for this variable must be integer, smallint, tinyint, decimal, or numeric. decimal and numeric must have a scale of zero.
- An *ordinality_column*, if one exists, is not nullable.

Note This default is different from the default value of create table.

- The nullable property of other columns is specified by the `{null | not null}` clause. The default is null.
- The current setting of set quoted_identifier applies to the clauses of an xmltable() expression. For example:
  - If set quoted_identifier is on, column names can be quoted identifiers, and string literals in *row_pattern*, *column_pattern*, and default literals must be surrounded with single quotation marks.
  - If set quoted_identifier is off, column names cannot be quoted identifiers, and string literals in *row_pattern*, *column_pattern*, and default literals can be surrounded with either single or double quotation marks.
- The general format of the option_string is described in "option_strings: general format," in *XML Services, Adaptive Server 15.0*.

xmltable() row and column patterns

- xmltable() row and column patterns are allowed to be only simple paths. Simple paths in XPath consist only of forward traversals using '/' and element/attribute names.
- If the *row_pattern* does not begin at the root level of the document specified by *xml_argument*, an exception is raised. The row pattern must begin at the root of the XML document.
- The row pattern expression cannot contain an XPath function.
- A column pattern must be a relative path.
- If the *row_pattern* specifies an XML function, an exception is raised. The row pattern cannot specify an XML function.

- If a *column_definition* does not specify a path, the default *column_pattern* is the *column_name* of the column definition. This default is subject to the case sensitivity of the server. For example, consider this statement:

  ```
  select * from xmltable(...columns name
  varchar(30),...)
  ```

  If the server is case-insensitive, this is equivalent to the following:

  ```
  select * from xmltable(...columns name varchar(30)
  path 'name',...)
  ```

  If the server is case-sensitive, the first statement is equivalent to:

  ```
  select * from xmltable
  (...columns name varchar(30)path 'NAME',...)
  ```

Generating the rows of the result table

- The result value of an xmltable() expression is a T-SQL table RT, defined as follows:
- RT has a row for each element in the XML sequence that results from applying the *row_pattern* to the *xml_argument*.
- The rows of RT have a column for each *column_definition*, with the *column_name* and datatype specified in the *column_definition*.
- If a *column_definition* is a *ordinality_column*, its value for the Nth row is the integer N.
- If a *column_definition* is a *regular_column*, its value for the Nth row corresponds to the following:
  - Let XVAL be the result of applying this XPath expression to the *xml_argument*:

    ```
    (row_pattern[N])/column_pattern/text()
    ```

  - If XVAL is empty, and the *column_definition* contains a default clause, the value of the column is that default value.

    If XVAL is empty and the *column_definition* specifies not null, an exception is raised.

    Otherwise, the value of the column is the null value.

  - If XVAL is not empty, and the datatype of the column is char, varchar, text, unitext, unichar, univarchar, or java.lang.String, de-entitize XVAL.
  - The value of the column is the result of:

    ```
    convert(datatype,XVAL)
    ```

CHAPTER 16

Relocated Joins

Adaptive Server version 15.0.2 introduces relocated joins which allow joins between local and remote tables to be relocated to a remote server. The remote system executes the join using a dynamically created proxy table referring back to the local table. With the remote system executing the join, a significant amount of network traffic is avoided.

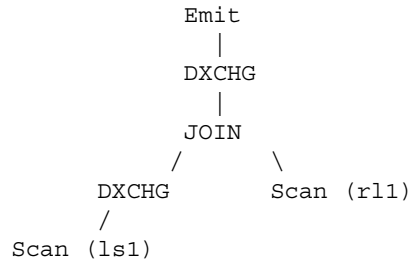
Using relocated joins

A join between local table ls1 and remote table rl1 results in the following query being sent to the remote server:

```
select a,b,c
from localserver.testdb.dbo.ls1 t1, rl1 t2
where t1.a = t2.a
```

The statement sent to the remote server contains a fully qualified reference back to the local table on the local system. The remote server will either dynamically create a temporary proxy table definition for this table, or use an existing proxy table with a matching mapping. The remote server will then execute the join and return the result set back to the local server.

The plan for this relocated join would appear as:



This plan has two distributed exchange operators present. The DXCHG above the scan of local table ls1 represents the cost of the remote server pulling results to the remote system. The DXCHG at the top of the plan represents the movement of the join result back to the local system.

Configuring relocated joins

Relocated joins must be specifically enabled for each remote server involved. This configuration must be done because the remote server must be able to connect back into the local server using Component Integration Services (CIS).

To configure relocated joins:

1 Use sp_serveroption to enable relocated joins to be sent to that remote server:

```
sp_serveroption  servername, "relocated joins",true
```

2 In order for Adaptive Server to establish a CIS connection back to the local server, verify the following on the remote server:

- The remote server has an interface entry for the local server
- A sysservers entry exists (added via sp_addserver)
- External logins have been configured

3 If you are using dynamically created proxy tables, proxy tables are created in tempdb when a relocated join is received. Enable ddl in tran to ensure that tempdb allows proxy tables:

```
sp_dboption tempdb,"ddl in tran",true
```

Dynamic proxy tables

The remote server requires a proxy table to process a relocated join. If a proxy table does not exist in advance, one is created automatically when the query is processed. When a relocated join query is received by the remote server, the current database is searched for an existing proxy table that has the same remote mapping as the table referenced in the relocated statement. If no proxy table exists with same mapping in the current database, then tempdb is searched. If no match is found there, then a proxy table is created dynamically in tempdb.

To avoid the overhead in dynamically creating a proxy table, Sybase recommends that you create proxy tables for each table that may be referenced in a relocated join.

Limitations

Relocated joins in Adaptive Server version 15.0.2 have the following limitations:

- Relocated joins are supported only with remote servers version 15.0.2 or higher.
- Queries referencing text/image will not be relocated to a remote server.
- Relocated joins are restricted to queries only.
- You cannot use a relocated join on client-side cursors.

CHAPTER 17

User-Defined SQL Functions

The 15.0.2 release of Adaptive Server includes the SQL functions described in this chapter:

User defined SQL functions overview

You can include these in a scalar function:

- declare statements to define data variables and cursors that are local to the function.
- Assigned values to objects local to the function (for example, assigning values to scalar and variables local to a table with select or set commands).
- Cursor operations that reference local cursors that are declared, opened, closed, and deallocated in the function.
- Control-of-flow statements.
- set options (only valid in the scope of the function).

Adaptive Server does not allow fetch statements in a scalar function that return data to the client. You cannot include :

- select or fetch statements that returns data to the client.
- insert, update, or delete statements.
- Utility commands, such as dbcc, dump and load commands.
- print statements
- Statement that references rand, rand2, getdate, or newid.

You can include select or fetch statements that assign values only to local variable.

create function

Description

Creates a user-defined function, which is a saved Transact-SQL routine that returns a specified value.

Syntax

```
create function [ owner_name. ] function_name
( [ { @parameter_name [as] parameter_datatype [ = default ] } [ ,...n ] ] )
returns return_datatype
[ with recompile] ]
as
[begin]
function_body
return scalar_expression
[end]
```

Parameters

- *owner_name* – the name of the user ID that owns the user-defined function. Must be an existing user ID.
- *function_name* – name of the user-defined function. Function names must conform to the rules for identifiers and must be unique within the database and to its owner. Function names cannot be the same as other Adaptive Server functions.
- *@parameter_name* – parameter in the user-defined function. You can declare one or more parameters in a create function statement. A function can have a maximum of 2,047 parameters. The value of each declared parameter must be supplied by the user when the function is executed, unless you define a default for the parameter.

  Specify a parameter name using an "at" sign (@) as the first character. The parameter name must conform to the rules for identifiers. Parameters are local to the function. You can use the same parameter names in other functions.

  If a parameter has a default value, the user must specify the keyword "default" when they call the function to get the default value.
- *Parameter_datatype* – the data type of the parameter. All scalar data types and Java abstract datatypes (ADTs) can be used as a parameter for user-defined functions. However, user-defined functions do not support the timestamp, text, image and unitext.

- *with recompile* – indicates that Adaptive Server never saves a plan for this function; instead, a new plan is created each time the function is first referenced in a SQL statement. Use *with recompile* when you expect the execution of this function is atypical, and it will require a new plan.
- To reference or invoke a user-defined function, specify the *owner_name.function_name* followed by parentheses (see the BONUS function in the "Examples" section below for an example). Specify expressions as arguments for all the parameters within the parentheses. You cannot specify the parameter names in the argument list when you invoke a function.You must supply argument values for all of the parameters, and the argument values must be in the same sequence in which the parameters are defined in the create function statement. When a function's parameter has a default value, you must specify the keyword "default" when calling the function to get the default value.
- You can invoke scalar-valued functions where scalar expressions are used, including computed columns and check constraint definitions.
- *return_datatype* – the return value of a scalar, user-defined function. It can be any of the scalar data types and Java ADTs except text, image, unitext and timestamp.
- scalar_expression – specifies the scalar value the scalar function returns.
- *function_body* – specifies a series of T-SQL statements, which together do not produce a side effect but define the value of the function. *function_body* is used only in scalar functions and multi-statement table-valued functions. In scalar functions, *function_body* is a series of T-SQL statements that evaluate to a scalar value.

Examples

This example creates a user-defined function named bonus:

```
create function BONUS(@salary int, @grade int, @dept_id int)
returns int
as
    begin
    declare @bonus int
    declare @cat int
    set @bonus = 0
    select  @cat = dept_cat from department where dept_id = @dept_id

    if (@cat < 10)
        begin
            set @bonus = @salary *15/100

        end
```

```
            else
                begin
                set     @bonus = @salary * 10/100
            end
    return @bonus
    end
```

Usage

If the owner of the user-defined function also owns all the database objects referenced inside, then all the other users who have execute permission on the function are automatically granted access permissions to all the referenced objects when they execute the function.

Permissions

By default, users with the sa and the dbo role have permission to run create function. Users with the sa or dbo role can grant create function permissions to other logins.

Owners of functions have execute permission on their functions. Other users do not have execute permissions unless execute permissions on the specific function are granted to them.

drop function

Definition

Removes one or more user-defined functions from the current database.

Syntax

drop function{ [*owner_name* .] *function_name* } [,...n]

Parameters

- *owner_name* – the name of the user ID that owns the user-defined function. Must be an existing user ID.
- *function_name* – name of the user-defined function to berremoved. Specifying the owner name is optional; the server name aand database name cannot be specified.

Examples

This example drops the bonus function:

```
drop function bonus
```

Usage

drop function drops scalar SQL user-defined functions from your current database.

Permissions

Permissions default to the function owner, and are not transferable. Users with the sa and the dbo roles can drop any object by specifying the owner.

CHAPTER 18

Using *instead of* Triggers

Views are commonly used to separate logical database schema from physical schema. This chapter discusses a feature that can be defined on a view to replace the standard action of an update, insert, or delete statement. The instead of trigger allows all views, including those that are not otherwise updatable, to be updated.

Overview

instead of triggers are special stored procedures that override the default action of a triggering statement (insert, update, and delete), and perform user-defined actions.

The instead of trigger, like the for trigger, executes each time a data modification statement executes on a specific view. A for trigger fires after an insert/update/delete statement on a table, and is sometimes called an after trigger. A single instead of trigger can apply to one specific triggering action:

```
instead of update
```

It can also apply to multiple actions, in which the same trigger executes all the actions listed:

```
instead of insert,update,delete
```

Like for triggers, instead of triggers use the logical inserted and deleted tables to store modified records while the trigger is active. Each column in these tables maps directly to a column in the base view referenced in the trigger. For example, if a view named V1 contains columns named C1, C2, C3, and C4, the inserted and deleted tables contain the values for all four columns, even if the trigger modifies only columns C1 and C3. Adaptive Server automatically creates and manages the inserted and deleted tables as memory-resident objects.

instead of triggers allow views to support updates, and allow implementation of code logic that requires rejecting parts of a batch, while allowing other parts to succeed.

An instead of trigger is fired only once per data modification statement. A complex query containing a while loop may repeat an update or insert statement many times, firing the instead of trigger each time.

Inserted and deleted logical tables

Deleted and inserted tables are logical (conceptual) tables. An inserted table is a pseudo-table containing rows with the inserted values of an insert statement, and a deleted table is a pseudo-table containing rows with the updated values (after image) of an update statement.The schema of the inserted and deleted tables is identical to that of the view for which the instead of trigger is defined; that is, the view on which a user action is attempted. The difference between these tables and the view is that all the columns of inserted and deleted tables are nullable, even when the corresponding column of the view is not. For example, if the view has a column of datatype char, and an insert statement provides a char value to be inserted, the inserted table has a datatype varchar for the corresponding column, and the input value is converted to varchar. However, trailing blanks are not truncated from the value when the value is added to the inserted table.

When a value is specified of a datatype different from that of the corresponding column in which it is inserted, the value is internally converted to the datatype of the column. If the conversion succeeds, the converted value is inserted into the table, but if the conversion fails, the statement is aborted. In this example, if a view selects an integer column from a table:

```
CREATE VIEW v1 AS SELECT intcol FROM t1
```

The following insert statement causes the instead of trigger on v1 to execute, because the value, 1.0, can be successfully converted to the integer value, 1:

```
INSERT INTO v1 VALUES (1.0)
```

However, this next statement causes an exception to be raised, and is aborted before the instead of trigger can execute:

```
INSERT INTO v1 VALUES (1.1)
```

The deleted and inserted tables can be examined by the trigger, to determine whether or how trigger action should be carried out, but the tables themselves cannot be altered by trigger action.

The deleted table is used with delete and update; the inserted table, with insert and update.

Note instead of triggers create the inserted and deleted tables as in-memory tables or worktables. for triggers generate the rows of inserted and deleted tables on the fly, by reading the transactional log, syslogs.

LOB datatypes

Values for columns in inserted or deleted tables that are Large OBject (LOB) datatypes are stored in memory, and can lead to significant memory usage if there are many rows in these tables and the rows contain large LOB values.

Triggers and transactions

Both the commands rollback trigger and rollback transaction are used for instead of triggers. rollback trigger rolls back the work done in any and all nested instead of triggers fired by the triggering statement. rollback transaction rolls back the work done in the entire transaction, up to the most recent savepoint.

Nesting and recursion

Like for triggers, instead of triggers can be nested to 16 levels. The current nesting level is stored in *@@nestlevel*. Nesting is on by default. A system administrator can use the configuration parameter allow nested triggers to turn trigger nesting on and off. If nested triggers are enabled, a trigger that changes a table containing another trigger executes the second trigger, which in turn can execute another trigger, and so forth, producing an infinite loop. In this case, processing ends when the nesting level is exceeded, and the trigger aborts. A rollback transaction in a trigger at any nesting level rolls back the effects of each trigger, and cancels the entire transaction. A rollback trigger affects only the nested triggers and the data modification statement causing the initial trigger to execute.

You can interleave nesting instead of and for triggers. For example, an update statement on a view with an instead of update trigger causes the trigger to execute. If the trigger contains a SQL statement updating a table with a for trigger defined on it, that trigger fires. The for trigger may contain a SQL statement that updates another view with an instead of trigger that then executes, and so forth.

Recursion

instead of and for triggers have different recursive behaviors, however. for triggers support recursion, while instead of triggers do not. If an instead of trigger references the same view on which the trigger was fired, the trigger is not called recursively. Rather, the triggering statement applies directly to the view; in other words, the statement is resolved as modifications against the base tables underlying the view. In this case, the view definition must meet all restrictions for an updatable view. If the view is not updatable, an error is raised.

For example, if a trigger is defined as an instead of update trigger for a view, the update statement executed against the same view within the instead of trigger does not cause the trigger to execute again. The update exercised by the trigger is processed against the view, as though the view did not have an instead of trigger. The columns changed by the update must be resolved to a single base table.

trigger syntax

instead of trigger

```
create trigger [owner.] trigger_name
    on [owner.]view_name
```

```
    instead of {insert, update, delete}
    as SQL_statements
```

instead of trigger with *update* clause

```
create trigger [owner.] trigger_name
    on [owner.]view_name
    instead of {insert, update, delete}
    as
        [if update (column_name)
            [{and | or} update (column_name)]...]
            SQL_statements
        [if update (column_name)
            [{and | or} update (column_name)]...
            SQL_statements] ...
```

drop *instead of* trigger statement

The syntax for dropping the instead of trigger is the same as for the for triggers.

Parameters

- *trigger_name* – is the name of the trigger, which must conform to the rules for identifiers and is unique in the database. To create another trigger of the same name, owned by a different user in the current database, specify the owner's name.
- *view_name* – is the name of the view on which to create the trigger.
- insert, update, delete – statements that can be included in any combination. delete cannot be used with an if update clause.
- *SQL_statements* – specify trigger conditions and actions. Trigger actions take effect when the user action (insert, update, delete) is attempted.
- if update – tests whether a specified column is included in the set list of an update statement, or is affected by an insert statement.

Examples

Example 1 These examples use instead of triggers to update union views.

```
CREATE TABLE EmployeeWest (
    empid                          int primary key,
    empname                      varchar(30),
    empdob                         datetime,
    region                             char(5)
        constraint region_chk
            check (region='West'))

CREATE TABLE EmployeeEast (
    empid                          int primary key,
    empname                      varchar(30),
    empdob                         datetime,
    region                             char(5)
        constraint region_chk
            check (region='East'))
```

```
CREATE VIEW Employees as
    SELECT * FROM EmployeeEast
    UNION ALL
    SELECT * FROM EmployeeWest

CREATE TRIGGER EmployeesInsertTrig on Employees
INSTEAD OF INSERT AS
BEGIN

    INSERT INTO EmployeeEast SELECT * FROM inserted
where
          region = "East"

    INSERT INTO EmployeeWest SELECT * FROM inserted
where
          region = "West"
END

--will insert the data into the EmployeeEast table
INSERT INTO Employees
       values (10, 'Jane Doe', '11/11/1967', 'East')

--will insert the data into the EmployeeWest table
INSERT INTO Employees
       values (11, 'John Smith', '01/12/1977', 'West')

--will insert multiple rows into EmployeeEast and
--EmployeeWest tables. Employee2 table includes
employees
--from both East and West.
INSERT INTO Employees SELECT * from Employee2
```

Example 2 These examples use instead of triggers to implement encrypted column support,storing data in the database in encrypted form without changing applications..

```
CREATE TABLE Employee_t (id int PRIMARY KEY, name
varchar(20),
          salary binary (64))
--where the id and name columns are stored unencrypted,
salary is
--encrypted and id is a primary key.

CREATE VIEW employee_v as select id, name, decrypt
(salary)
FROM employee_t

CREATE TRIGGER EmployeeInsert
```

```
ON employee_v
INSTEAD OF INSERT
AS
BEGIN
    INSERT employee_t SELECT id, name, encrypt (salary)
    FROM inserted
END

CREATE TRIGGER employeeUpdate
ON employee_v
INSTEAD OF UPDATE
AS
BEGIN
    DELETE FROM employee_t WHERE id IN (SELECT id FROM
    deleted)
    INSERT employee_t SELECT id, name, encrypt (salary)
    FROM inserted
END

CREATE TRIGGER employeeDelete
ON employee_v
INSTEAD OF DELETE
AS
BEGIN
    DELETE FROM employee_t WHERE id IN (SELECT id FROM
    deleted)
END
```

Usage

- The default value of *owner_name* is the current user.
- If you use an *owner_name* to qualify a trigger, you must explicitly qualify the *view_name* the same way.
- You cannot use a variable for a *trigger_name*.
- If more than one view of the same name exists in the database, specify the owner's name.
- If multiple trigger actions are specified in *SQL_statements*, group them with BEGIN and END.

Restrictions

- You cannot create instead of triggers on a table, although triggers can reference tables. You can only create instead of triggers on views.
- You can define only one instead of trigger for each insert, update, or delete statement on a view. A view can have one trigger defined for multiple operations, or separate triggers for each operation.
- Each new trigger created on a view for the same operation overwrites the previous trigger.

- A trigger cannot apply to more than one view.
- instead of triggers are not allowed on updatable views using the with check option.
- The create trigger statement must be the first statement in the batch. All other statements that follow in that batch are treated as part of the definition of the create trigger statement.
- Permission to create triggers defaults to the view owner, who cannot transfer it to other users.
- Triggers are database objects, and their names must follow the rules for identifiers.
- You can create a trigger only in the current database, although a trigger can reference objects outside of the current database.
- Enabling and disabling the instead of trigger is not supported.
- If a trigger references table names, column names, or view names that are not valid identifiers, you must set quoted_identifier on before the create trigger command, and enclose each such name in double quotation marks. The quoted_identifier option does not need to be on when the trigger fires; bracketed identifiers work as well.
- Using the set cursor rows command with **client cursors,** cursors declared through Open Client calls or embedded SQL, may prevent **positioned** delete and update from firing an instead of trigger. A positioned update statement is a SQL update statement that contains the where current of <*cursorname*> clause to update only the row upon which the cursor, <*cursorname*>, is currently positioned.
- Joins are not allowed in searched delete and update statements that would fire an instead of trigger.
- positioned delete and update on cursors defined with joins does not fire an instead of trigger.

  A positioned delete is a SQL delete statement containing a where current of <*cursorname*> clause to delete only the row upon which the cursor, <*cursorname*>, is currently positioned.

  A positioned update is a SQL update statement that contains the where current of <*cursorname*> clause to update only the row upon which the cursor, <*cursorname*>, is currently positioned.

- For positioned delete and update statements that fire an instead of trigger, the instead of trigger must exist when the cursor is declared.

Note instead of triggers on tables are not currently implemented.

Permissions

instead of triggers have the same permission requirements as for triggers: to create a view with instead of triggers, permission for insert/update/delete for the view, not the underlying tables, must be granted to the user.

instead of insert triggers

instead of insert triggers are defined on a view to replace the standard action of the insert statement. Usually, this trigger is defined on a view to insert data into one or more base tables.

Columns in the view select list can be nullable or not nullable. If a view column does not allow nulls, a SQL statement that inserts rows into the view must provide a value for the column. In addition to valid non-null values, an explicit value of null is also accepted for a non-nullable column of the view. view columns allow nulls if the expression defining the view column includes such items as:

- References to any base table column that allows nulls
- Arithmetic operators
- References to functions
- CASE with a nullable subexpression
- NULLIF

The stored procedure sp_help reports which view columns allow nulls.

An insert statement referencing a view with an instead of insert trigger must supply values for every view column that does not allow nulls. This includes view columns that reference columns in the base table for which input values cannot be specified, such as:

- Computed columns in the base table
- Identity columns in the base table for which identity insert is OFF.

If the instead of insert trigger contains an insert statement against the base table using data in the inserted table, the insert statement must ignore the values for these types of columns by not including them in the select list of the statement.The insert statement against the view can generate dummy values for these columns, but the insert statement in the instead of insert trigger ignores those values and Adaptive Server supplies the correct values.

Example

An insert statement must specify a value for a view column that maps to an identity or computed column in a base table. However, it can supply a placeholder value. The insert statement in the instead of trigger that inserts the values into the base table is written to ignore the supplied value.

For example, these statements create a table, view, and trigger that illustrate the process:

```
CREATE TABLE BaseTable
(PrimaryKey         int IDENTITY
Color               varchar (10) NOT NULL,
Material            varchar (10) NOT NULL,
TranTime            timestamp
)
---------------------
--Create a view that contains all columns from the base
table.
CREATE VIEW  InsteadView
AS SELECT PrimaryKey, Color, Material, TranTime
FROM BaseTable
---------------------
Create an INSTEAD OF INSERT trigger on the view.
CREATE TRIGGER InsteadTrigger on InsteadView
INSTEAD OF INSERT
AS
BEGIN
    --Build an INSERT statement ignoring
    --inserted.PrimaryKey and
    --inserted.TranTime.
INSERT INTO BaseTable
    SELECT Color, Material
    FROM inserted
END
```

An insert statement that refers directly to BaseTable cannot supply a value for the PrimaryKey and TranTime columns. For example:

```
--A correct INSERT statement that skips the PrimaryKey
--and TranTime columns.
INSERT INTO BaseTable (Color, Material)
          VALUES ('Red', 'Cloth')

--View the results of the INSERT statement.
SELECT PrimaryKey, Color, Material, TranTime
FROM BaseTable

--An incorrect statement that tries to supply a value
--for the PrimaryKey and TranTime columns.
INSERT INTO BaseTable
   VALUES (2, 'Green', 'Wood', 0x0102)

INSERT statements that refer to InsteadView, however,
   must supply a value for PrimaryKey:

--A correct INSERT statement supplying a dummy value for
--the PrimaryKey column. A value for TranTime is not
--required because it is a nullable column.
INSERT INTO InsteadView (PrimaryKey, Color, Material)
       VALUES (999, 'Blue', 'Plastic')
--View the results of the INSERT statement.
SELECT PrimaryKey, Color, Material, TranTime
FROM InsteadView
```

The inserted table passed to InsteadTrigger is built with a non-nullable PrimaryKey column; therefore, the insert statement referencing the view must supply a value for this column. The value 999 is passed in to InsteadTrigger, but the insert statement in InsteadTrigger does not select inserted.PrimaryKey therefore, the value is ignored. The row actually inserted into BaseTable has 2 in PrimaryKey and an Adaptive Server-generated timestamp value in TranTime.

If a not null column with a default definition is referenced in a view with an instead of insert trigger, any insert statement referencing the view must supply a value for the column. This value is required to build the inserted table passed to the trigger. A convention is required for a value that signals the trigger that the default value should be used. A possible convention is to supply an explicit null value for the not null column in the insert statement. The instead of insert trigger can ignore the explicit null when inserting into the table upon which the view is defined, causing the default value to be inserted into the table. For example:

```
--Create a base table with a not null column that has
--a default
CREATE TABLE td1 (coll int DEFAULT 9 NOT NULL, col2 int)
```

```
--Create a view that contains all of the columns of
--the base table.
CREATE VIEW vtd1 as select * from td1
--create an instead of trigger on the view
CREATE TRIGGER vtd1insert on vtd1 INSTEAD OF INSERT AS
BEGIN
    --Build an INSERT statement that inserts all rows
    --from the inserted table that have a NOT NULL value
    --for col1.
    INSERT INTO td1 (col1,col2) SELECT * FROM inserted
    WHERE col != null

    --Build an INSERT statement that inserts just the
    --value of col2 from inserted for those rows that
    --have NULL as the value for col1 in inserted. In
    --this case, the default value of col1 will be
    --inserted.
    INSERT INTO td1 (col2) SELECT col2 FROM inserted
    WHERE col1 = null
END
```

The deleted table in an instead of insert trigger is always empty.

instead of update trigger

Usually, the instead of update trigger is defined on a view, to modify data in one or more base tables.

In update statements that reference views with instead of update triggers, any subset of the columns in the view can appear in the set clause of the update statement, whether they are non-nullable columns or not.

Even non-updatable view columns (those referencing columns in the base table for which input values cannot be specified) can appear in the set clause of the update statement. Non-updatable columns include:

- Computed columns in the base table
- Identity columns in the base table, for which identity insert is set to off
- Base table columns of the timestamp datatype.

Usually, when an update statement referencing a table attempts to set the value of a computed, identity, or timestamp column, an error is generated, because Adaptive Server must generate the values for these columns. However, if the update statement references a view with an instead of update trigger, the logic defined in the trigger can bypass these columns and avoid the error. To do so, the instead of update trigger must not try to update the values for the corresponding columns in the base table. This is done by excluding the columns in the set clause of the update statement from the definition of the trigger.

This solution works because an instead of update trigger does not need to process data from the inserted columns that are not updatable. The inserted table contains the values of columns not specified in the set clause as they existed before the update statement was issued. The trigger can use an if update (column) clause to test whether a specific column has been updated.

instead of update triggers should use values supplied for computed, identity, or timestamp columns only in where clause search conditions. The logic that an instead of update trigger on a view uses to process updated values for computed, identity, timestamp, or default columns is the same as the logic applied to inserted values for these column types.

The inserted and deleted tables each contain a row for every row that qualifies to be updated, in an update statement on a view with an instead of update trigger. The rows of the inserted table contain the values of the columns after the update operation, and the rows of the deleted table contain the values of the columns before the update operation.

instead of delete trigger

instead of delete triggers can replace the standard action of the delete statement. Usually, the instead of delete trigger is defined on a view to modify data in base tables.

delete statements do not specify modifications to existing data values, only the rows to be deleted. A delete table is a pseudo-table containing rows with the deleted values of a delete statement, or the pre-updated values (before image) of an update statement.The deleted table sent to an instead of delete trigger contains an image of the rows as they existed before the delete statement was issued. The format of the deleted table is based on the format of the select list defined for the view, with the exception that all not null columns be converted to nullable columns.

The inserted table passed to a delete trigger is always empty.

searched and positioned *update* and *delete*

update and delete statements can be either searched or positioned commands. A searched delete is a SQL delete statement containing an optional predicate expression in the where clause, to qualify rows to be deleted. A searched update is a SQL update statement containing an optional predicate expression in the where clause, to qualify rows to be updated.

An example of a searched delete statement is:

```
DELETE myview WHERE myview.col1 > 5
```

This statement is executed by examining all the rows of myview, and applying the predicate (myview.col1 > 5) specified in the where clause to determine which rows should be deleted.

Joins are not allowed in searched update and delete statements. To use the rows of another table to find the qualifying rows of the view, use a subquery. For example, this statement is not allowed:

```
DELETE myview FROM myview, mytab
    where myview.col1 = mytab.col1
```

But the equivalent statement, using a subquery, is allowed:

```
DELETE myview WHERE col1 in (SELECT col1 FROM mytab)
```

Positioned update and delete statements are executed only on the result set of a cursor, and affect only a single row. For example:

```
DECLARE mycursor CURSOR FOR SELECT * FROM myview
OPEN mycursor
FETCH mycursor
DELETE myview WHERE CURRENT OF mycursor
```

The positioned delete statement deletes only the row of myview on which mycursor is currently positioned.

If an instead of trigger exists on a view, it always executes for a qualifying searched delete or update statement; that is, a statement without joins. For an instead of trigger to execute on a positioned delete or update statement, the following two conditions must be met:

- The instead of trigger exists when the cursor is declared; that is, when the command declare cursor is executed.
- The select statement that defines the cursor can access only the view; for example, the select statement contains no joins, but it can access any subset of the view columns.

The instead of trigger also executes when positioned delete or update statements are executed against scrollable cursors, also. However, instead of triggers do not fire in one case, when using a **client cursor** and the command set cursor rows.

Client cursors

A client cursor is declared and fetched in an application using the Open Client™ library functions for cursor processing. The Client library functions can retrieve multiple rows from Adaptive Server in a single fetch command, and buffer these rows, returning one row at a time to the application on subsequent fetch commands, without having to retrieve any more rows from Adaptive Server until the buffered rows are all read. By default, Adaptive Server returns a single row to the Open Client library functions for each fetch command it receives. However, the command set cursor rows can change the number of rows Adaptive Server returns.

Positioned update and delete statements for client cursors, for which set cursor rows is not used to increase the number of rows returned per fetch, cause an instead of trigger to execute. However, if set cursor rows increases the number of rows returned per fetch command, an instead of trigger executes only if the cursor is not marked read-only during the internal processing of declare cursor. For example:

```
--Create a view that is read-only (without an instead
--of trigger) because it uses DISTINCT.
CREATE VIEW myview AS
SELECT DISTINCT (col1) FROM tab1

--Create an INSTEAD OF DELETE trigger on the view.
CREATE TRIGGER mydeltrig ON myview
INSTEAD OF DELETE
AS
BEGIN
    DELETE tab1 WHERE col1 in (SELECT col1 FROM deleted)
END

Declare a cursor to read the rows of the view
DECLARE cursor1 CURSOR FOR SELECT * FROM myview

OPEN cursor1
```

```
FETCH cursor1

--The following positioned DELETE statement will
--cause the INSTEAD OF TRIGGER, mydeltrig, to fire.
DELETE myview WHERE CURRENT OF cursor1

--Change the number of rows returned by ASE for
--each FETCH.
SET CURSOR ROWS 10 FOR cursor1

FETCH cursor1

--The following positioned DELETE will generate an
--exception with error 7732: "The UPDATE/DELETE WHERE
--CURRENT OF failed for cursor 'cursor1' because
--the cursor is read-only."
DELETE myview WHERE CURRENT OF cursor1
```

By using set cursor rows, a disconnect between the position of the cursor in Adaptive Server and in the application is created: Adaptive Server is positioned on the last row of the 10 rows returned, but the application can be positioned on any one of the 10 rows, since the Open Client library functions buffer the rows and scroll through them without sending information to Adaptive Server. Because Adaptive Server cannot determine the position of cursor1 when the positioned delete statement is sent by the application, the Open Client library functions also send the values of a subset of the columns of the row where cursor1 is positioned in the application. These values are used to convert the positioned delete into a searched delete statement. This means that if the value of col1 is 5 in the current row of cursor1, a clause such as `'where col1 = 5'` is used by Adaptive Server to find the row.

When a positioned delete is converted to a searched delete, the cursor must be updatable, just as for cursors on tables and on views without instead of triggers. In the example above, the select statement that defines cursor1 is replaced by the select statement that defines myview:

```
DECLARE cursor1 CURSOR FOR SELECT * FROM myview
```

becomes:

```
DECLARE cursor1 CURSOR FOR SELECT DISTINCT (col1)
FROM tab1
```

Because of the distinct option in the select list, cursor1 is not updatable; in other words, it is read-only.. This leads to the 7732 error when the positioned delete is processed.

If the cursor that results from replacing the view by its defining select statement is updatable, the instead of trigger fires, whether set cursor rows is used or not. Using set cursor rows does not prevent an instead of trigger from firing in other types of cursors Adaptive Server supports.

Getting information about triggers

Information about instead of triggers is stored as it is for for triggers.

- The definition query tree for a trigger is stored in sysprocedures.
- Each trigger has an identification number (object ID), stored in a new row in sysobjects. The object ID of the view to which the trigger applies is stored in the deltrig, instrig, and updtrig columns of the sysobjects row for the trigger. The object ID of the trigger is stored in the deltrig, instrig, or putrid columns in the sysobjects row of the view to which the trigger applies.
- To display the text of a trigger stored in syscomments, use sp_helptext. If the system security officer has reset the column parameter allow select on syscomments.text with sp_configure, you must be the creator of the trigger or a system administrator to view the text of the trigger through sp_helptext.
- Use sp_help to obtain a report on a trigger. sp_help reports instead of trigger as the *object_type* of an instead of trigger.
- Use sp_depends to report on the views referenced by an instead of trigger. sp_depends *view_name* reports the trigger name and its type as instead of trigger.

CHAPTER 19

Changes to *ddlgen*

This chapter describes the Adaptive Server 15.0.2 changes to the ddlgen utility. For details on ddlgen, see the *Utility Guide*.

Using the *-P* password parameter

In Adaptive Server prior to version 15.0.2, if you did not include the -P password parameter in your ddlgen statement, ddlgen returned an error. In 15.0.2, if you do not include the -P parameter in your ddlgen statement, ddlgen returns a prompt that asks you to specify a password.

Using *ddlgen* for encrypted columns

This section discusses encrypted column support in ddlgen, a command-line utility used to generate data definition language (DDL) scheme definitions of objects in Adaptive Server.

Pre-15.0.2 *ddlgen* support

The ddlgen utility supports both pre-15.0.2 and 15.0.2 encryption.

Generating DDL for an encryption key in a database

To generate DDL for an encryption key “ssn_key” in a database called “SampleKeysDB,” the syntax is:

```
ddlgen -Usa -P -Sserver -TEK -NSampleKeysDB.dbo.ssn_key
```

Where:

- EK – is the encrypted key type
- SampleKeysDB.dbo.ssn_key – is the fully qualified name for the encrypted key

ddlgen generates the following using this syntax:

```
------------------------------------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------
print 'ssn_key'

create encryption key SampleKeysDB.dbo.ssn_key for AES
with keylength 128
init_vector random
go
```

Generating DDL to synchronize encryption keys across servers

To synchronize encryption keys across servers for data movement, use the ddlgen -XOD extended option to generate the create encryption key command that specifies the key's encrypted value as represented in sysencryptkeys, such as in this sample:

ddlgen -Usa -P -S*server* -TEK -NSampleKeysDB.dbo.ssn_key -XOD

The output from this ddlgen command would be:

```
------------------------------------------------------------------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------------------------------
print 'ssn_key'

create encryption key SampleKeysDB.dbo.ssn_key for AES
with keylength 128
passwd 0x00009EECD959E00095A401
init_vector random
keyvalue
0x11D6B8AA6ACB0C0490A363791531DD6E6728C800FDD1A13BFF795FA22726C16101
keystatus 32
go
```

Adaptive Server 15.0.2 *ddlgen* support

The ddlgen utility also supports 15.0.2 encryption keys.

Generating DDL without specifying the -XOD flag

For 15.0.2 encryption, two things can happen if users do not specify the -XOD flag in ddlgen:

- If users **did not** specify a password when the encryption key was created, ddlgen generates DDL with no password.
- If users specified a password when the encryption key was first created, ddlgen generates the default password of 'password'. This is similar to what ddlgen does for roles and login passwords, and its output looks similar to the following:

```
------------------------------------------------------------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------------------------
print 'ssn_key'

--The DDL is generated with a default password – 'password' as
--a password was specified when this key was created.

create encryption key SampleKeysDB.dbo.ssn_key for AES
with keylength 128
passwd 'password'
init_vector random
go
```

Generating DDL with the -XOD flag

In 15.0.2, when a user specifies "-XOD" in the ddlgen command, ddlgen generates DDL that includes a system encryption password (if it has been set and DDL is generated for a key encrypted with a system encryption password) and DDL for keys.

The following syntax generates a system encryption password:

```
ddlgen -Usa -P -Sserver -TEK -NsampleKeysdb.dbo.ek1 -XOD
```

The output for the command is:

```
-- System Encryption Password

use SampleKeysDB
go

sp_encryption 'system_encr_passwd',
'0x8e050e3bb607225c60c7cb9f59124e99866ca22e677b2cdc9a4d09775850f4721',
NULL, 2, 0
go

------------------------------------------------------------------------
-- DDL for EncryptedKey 'ek1'
------------------------------------------------------------------------

print '<<<<< CREATING EncryptedKey - "ek1" >>>>>'
go
```

```
create encryption key SampleKeysDB.dbo.ek1 for AES
with keylength 128
passwd 0x0000C7BC28C3020AC21401
init_vector NULL
keyvalue
0xCE74DB1E028FF15D908CD066D380AB4AD3AA88284D6F7742DFFCADCAABE4100D01
keystatus 32
go
```

Note When migrating keys from a source to a target server using ddlgen, set the system encryption password to NULL (if it exists) in the target server if you want to run the ddlgen output (from the source server) for encryption keys generated using "-XOD" parameter. Failure to do this results in errors when you try to execute the ddlgen output against the target server.

Key copy support

The ddlgen utility also generates DDL for key copies along with the DDL for base key. For example, the following syntax would generate DDL for "ssn_key" and its key copies:

ddlgen -Usa -P -S*server* -TEK -NSampleKeysDB.dbo.ssn_key

The output from ddlgen would look like:

```
------------------------------------------------------------------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------------------------------
print 'ssn_key'

--The DDL is generated with a default password – 'password' as
--a password was specified when this key was created.

create encryption key SampleKeysDB.dbo.ssn_key for AES
with keylength 128
passwd 'password'
init_vector random
go

print 'Key Copies for ssn_key'

-- Generating DDL for Key Copies for 'ssn_key'
```

```
alter encryption key 'ssn_key'
with passwd 'password'
add encryption with passwd 'passwd'
for user 'dbo'.
```

If you include the -XOD flag, the DDL for key copy would look like:

```
alter encryption key SampleKeysDB.dbo.ssn_key add encryption
with keyvalue
0x84A7360AA0B28801D6D4CBF2F8219F634EE641E1082F221A2C58C9BBEC9F49B501
passwd 0x000062DF4B8DA5709E5E01
keystatus 257
for user 'user1'
go
```

EKC encryption key copy filter

With Adaptive Server 15.0.2, ddlgen supports the EKC (encryption key copy) extended type for its -F filter argument, to suppress the generation of key copies for encryption keys.

This example uses -FEKC to avoid creating DDL for key copies when generating DDL for the "ssn_key" encryption key:

ddlgen -Usa -P -S*server* -TEK -NSampleKeysDB.dbo.ssn_key -FEKC

The output from ddlgen would look like:

```
------------------------------------------------------------------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------------------------------
print 'ssn_key'

--The DDL is generated with a default password – 'password' as
--a password was specified when this key was created.

create encryption key SampleKeysDB.dbo.ssn_key for AES
with keylength 128
passwd 'password'
init_vector random
go
```

Create table DDL

ddlgen can generate decrypt_default statements (if set for an encrypted column) along with DDL of a table.

The following example issues a ddlgen command on a table called employee which has an "ssn" column that is encrypted with encryption key "ssn_key," and a decrypt default value that is set to "100":

```
ddlgen -Usa -P -Sserver -TU -Nemployee
```

The DDL output from the command is:

```
create table employee (
  ssn            int       not null  encrypt with ssn_key decrypt_default 100 ,
  last_name      int       not null ,
  first_name     int       not null
)
lock allpages
 on 'default'
go
```

CHAPTER 20

Changes to System Procedures, Functions, and Commands

This chapter describes new and changed system procedures, functions, and commands for Adaptive Server version 15.0.2.

Changed commands

This section discusses changed commands in Adaptive Server version 15.0.2.

disk init and disk reinit

disk init and disk reinit display a warning message if you attempt to create a block device on a platform that Sybase recommends that you not use block device.

Sybase recommends that you use block devices as a database device only on the HP-UX, Windows, and Linux platforms.

New parameter for *disk init*

Description

skip_alloc is a boolean parameter for the disk init command. It is supported for devices created on non-Windows file systems and on Windows raw systems. When skip_alloc is set to be "true." it allows user to avoid initialization of pages with zeros. The default of skip_alloc is false.

Syntax

The disk init syntax is**:**

```
disk init name="device name",
physname="physical name",
skip_alloc={true|false},
size=number_of_blocks
```

Example

This example does not initialize pages with zeroes.

```
disk init name="d2",
physname="/usr/sybase/devices/d3.dat",
skip_alloc="true",
size="10G"
```

Adaptive Server does not allocate space during disk initialization if skip_alloc is set to true.

Usage

Use skip_alloc to expedite crash recovery on non-NT file systems and on NT raw systems. Also, using skip_alloc with the directio feature creates device faster and improves durability of updates. Regardless of space availability, skip_alloc always prints a warning message to users saying they need to make sure Adaptive Server has the required space for future use.

New parameter for *disk resize*

Description

skip_alloc is a boolean parameter for the disk resize command. It is supported for devices created on non-NT file systems and on NT raw systems. When skip_alloc is set to be true it allows user to avoid initialization of pages with zeros. The default of skip_alloc is false.

Syntax

The disk resize syntax is:

```
disk resize name="device_name"
size=additional_space,
skip_alloc={true|false}
```

Example

This example does not initialize pages with zeroes for additional space.

```
disk resize name="dl", size="5G", skip_alloc="true"
```

With skip_alloc set to true Adaptive Server does not allocate space during disk initialization or extension. Ensure that the allocated space is available for Adaptive Server at the time of allocation.

Usage

Use skip_alloc to expedite crash recovery on non-NT file systems and on NT raw systems. Also, using skip_alloc in conjunction with directio creates devices faster and improves durability of updates.

Changes to set command

Compile-time changes for some set parameters

Adaptive Server 15.0.2 changes the compile-time behavior for some abstract plan set parameters when you use them to create stored procedures or run them in Transact-SQL batches.

In previous versions of Adaptive Server, the set parameters took effect after the stored procedure was executed or recompiled. Adaptive Server 15.0.2 allows you to use optimizer set parameters at compile time to affect the optimizer in stored procedures or batches.

Note This changed behavior may effect the composition of the result set. Sybase recommends that you review the result set created by the 15.0.2 versions of the set parameters before using them in your production systems.

You must reset the set parameter before returning from the stored procedure or the execution of subsequent stored procedures may be affected. If you intend to propogate this change to subsequent stored procedures, use export_options parameter.

Adaptive Server 15.0.2 changes the compile-time behavior for these parameters:

- distinc_sorted
- distinct_sorting
- distinct_hashing
- group_sorted
- group_hashing
- bushy_space_search

- parallel_query
- order_sorting
- nl_join
- merge_join
- hash_join
- append_union_all
- merge_union_all
- merge_union_distinct
- hash_union_distinct
- store_index
- index_intersection
- index_union
- multi_table_store_ind
- opportunistic_distict_view
- advanced_aggregation
- replicated_partition
- group_inserting
- ase125_primed
- auto_query_tuning
- query_tuning_mem_limit
- query_tuning_time_limit
- set plan optgoal

set advanced_aggregation

set advanced_aggregation enables and disables advanced aggregation at the session level.

```
set advanced_aggregation on/off
```

For more information, see “Eager and Lazy Aggregation.”

Changes to set and dump transaction

In previous versions of Adaptive Server you could not supply string and char_variable options with set and dump transaction. Adaptive Server version 15.0.2 allows these commands to include string and char_variable parameters. However, you cannot supply string and char_variable options to execute begin transaction, commit, connect to, declare cursor, rollback, dbcc, use or nested execute commands.

set ... set_switch

Description

Allows you to set trace flags and switch names locally and server-wide.

Syntax

set switch [serverwide] <on | off><*trace_flag* ,[*trace_flag*]>
[with <*option* [, *option*]>

Parameters

- serverwide – optional and will set a switch serverwide ON or OFF. The default is session-specific.
- on – trace flags are switched on.
- off – trace flags are switched off.
- *trace_flag* – a sequence of numbers (the old traceflag numbers) and/or switch names.
- *option* – an optional sequence of switch options. Valid values are:
  - override – this option is necessary to enable a non-documented switch names or trace flags
  - no_info – this option is used to surpress any informational warnings

Examples

```
set switch serverwide on 110 with override, no_info
set switch on 3601, 3604
set switch off 3601
```

SQL UDF

SQL UDF is not currently supported with create proxy table, create table at remote server, or alter table.

Note The execution of SQL Functions requires this syntax:

username.functionname()

SQL UDF permission check

When a function is created, Adaptive Server checks to see if it is a SQL UDF or a SQLJ UDf. If it is a SQLJ UDF, Adaptive Server checks for “sa” permissions. If it is a SQL Function Adaptive Server checks for “create function” priveledges.

Changed utilities

ddlgen

The ddlgen utility also supports 15.0.2 encryption keys. See “Changes to ddlgen” on page 243 for details.

New and changed system procedures

New system procedures

sp_downgrade

Description

Validates readiness for downgrade to an earlier 15.0.x release. Also downgrades the system catalog changes Adaptive Server 15.0.2 modified.

Syntax

```
sp_downgrade @cmd = {'prepare' | 'downgrade' | 'help',}
    @toversion = 'n'[, @verbose = 0 | 1][, @override = 0 | 1]
```

Parameters

- prepare – use first to validate readiness of Adaptive Server 15.0.2 for downgrade.
- downgrade – use after prepare parameter when ready to proceed with the act of downgrading to a previously installed 15.x version of Adaptive Server. Server must be in single user mode. (started with -m option)
- toversion – can be 15.0 or 15.0.1 written “150” or “15.0”, “1501” or “15.0.1”
- verbose – specifies verbosity. Valid options are 0 (for no) or 1 (for yes).
- override – specifies whether to skip databases that are not writable at this time. Valid options are 0 (for no) or 1 (for yes).

Examples

```
00:00000:00006:2007/04/25 17:39:29.63 server  Starting downgrading ASE.
00:00000:00006:2007/04/25 17:39:29.64 server  Downgrade : Downgrading login
passwords.
00:00000:00006:2007/04/25 17:39:29.64 server  Downgrade : Starting password
downgrade.
00:00000:00006:2007/04/25 17:39:29.64 server  Downgrade : Removed sysattributes rows.
00:00000:00006:2007/04/25 17:39:29.64 server  Downgrade : Updated 0 passwords.
00:00000:00006:2007/04/25 17:39:29.65 server  Downgrade : Removed columns in
syslogins - lastlogindate, crdate, locksuid, lock reason, lockdate are removed.
00:00000:00006:2007/04/25 17:39:29.65 server  Downgrade : Truncated password lengths.
00:00000:00006:2007/04/25 17:39:29.65 server  Downgrade : Successfully completed
password downgrade.
00:00000:00006:2007/04/25 17:39:29.65 server  Downgrade : Marking stored procedures
for recompilation.
00:00000:00006:2007/04/25 17:39:31.31 server  Downgrade : Dropping Sysoptions system
table.
00:00000:00006:2007/04/25 17:39:31.33 server  Downgrade : Setting master database
minor upgrade version.
00:00000:00006:2007/04/25 17:39:31.33 server  Downgrade : Setting user databases
```

```
minor upgrade version.
```

Usage

Use to revert to the previously installed Adaptive Server 15.0.x release. At this time it is not possible to revert to Adaptive Server 15.0.2.

sp_spaceusage

Description

Reports the space usage for a table, index, or transaction log and estimates the amount of fragmentation for tables and indexes in a database. The estimates are computed using an average row-length for data and index rows, and the number of rows in a table. You can archive the space usage and fragmentation data for future reporting and trends analysis. sp_spaceusage supports a number of actions, including help, display, archive and report, to indicate the current Adaptive Server space usage.

Syntax

sp_spaceusage supports a variety of operations.

```
sp_spaceusage
```

The “help” action syntax:

```
sp_spaceusage   'help'[, 'all']

sp_spaceusage 'help' [, {'display' |  'display summary'
            | 'report' | 'report summary' | 'archive'}
            [, {'table' | 'index' | 'tranlog'}]]
```

The “display” action syntax:

```
sp_spaceusage 'display summary [using unit= {KB | MB | GB | PAGES}
]',
      {'table' | 'index'}, name
      [,where_clause [,order_by[,command ] ] ]

sp_spaceusage 'display [using unit= {KB | MB | GB | PAGES} ]',
      {'table' | 'index'}, name
      [,select_list
      [,where_clause [,order_by[,command] ] ] ]

sp_spaceusage 'display [using unit={KB | MB | GB | PAGES} ]',
      'tranlog' [, name[,select_list[,where_clause [,order_by]]]]
```

The “archive” action syntax:

```
sp_spaceusage 'archive [ using_clause ]',
      {'table' | 'index'}, name[,where_clause[,command] ]

sp_spaceusage 'archive [ using_clause ]',
      'tranlog' [,name[,where_clause] ]
```

The “report” action syntax:

```
sp_spaceusage 'report summary [ using_clause ]',
    {'table' | 'index'}, name
    [,where_clause [,order_by[,from_date [,to_date]]]]

sp_spaceusage 'report [ using_clause ]',
    {'table' | 'index'}, name
    [,select_list[,where_clause [,order_by[,from_date [,to_date]]]]]

sp_spaceusage 'report [ using_clause ]',
    'tranlog' [, name
    [,select_list[,where_clause [,order_by
    [,from_date [,to_date]]]]]

using_clause = USING using_item [, using_item ...]

using_item = { unit={ KB | MB | GB | PAGES }
    | dbname=database_name | prefix=string }
```

Parameters

- help – displays the entire sp_spaceusage syntax. help *action* displays the syntax for individual actions supported..
- display – displays current space usage information for the specified objects.
- display summary – displays a summary of current space usage information for the specified objects.
- archive – archives the space usage report to a table. If the archive table does not already exist, sp_spaceusage creates one. New data is appended to existing data. You can specify a prefix for the archive table name and the database in which the archive table resides with the *using* clause.
- report – reports the space usage information for the specified objecs from previously archived data. The output is same as the display action. Include the optional using clause to specify the archive table.
- report summary – reports a summary of space usage information for the specified objecs from previously archived data. The output is same as the display summary action. Include the optional using clause to specify the archive table.
- *using_item* – specifies the unit, archive database name, and prefix string for the archive table. You can use a *unit* size of kilobytes (KB), megabytes (MB), gigabytes (GB), and pages. By default *unit* size is KB, the current database is the archive database, and no prefix string is assumed.

- *name* – is the name of the entity. Depending on the entity type, you can include multipart names such as *owner_name.table_name*, or *owner_name.table_name.index_name*. For the entity type tranlog, the name must be syslogs or NULL. Pattern specifiers are allowed for each part of a multipart name to support reporting on multiple objects in one pass.
- *select_list* – is the comma-separated list of columns to select in the output columns for the display and report actions. Use `*` to include all columns in the output. Columns can be renamed using the alias=*name* notation.
- *where_clause* – is the filter to apply to the result set. Use with the display, report, or archive actions to selectively filter unnecessary data.
- *order_by* – returns query results in the specified columns in sorted order.
- *command* – command run on the entity selected (table, column, or so on) prior to gathering the space usage information for qualifying objects. The following commands are supported: update statistics, update table statistics, and update index statistics.
- *from_date* – specifies beginning of the time range you are interested in.
- *to_date* – specifies end of the time range you are interested in.

Examples

Example 1 Displays a brief description, syntax, and usage information for the display action:

```
sp_spaceusage 'help', 'display'

Display the space usage information for an entity in the current database.

Usage:
sp_spaceusage 'display', {'table'|'index'}, <name>
              [,<select_list> [,<where_clause> [,<order_by> [,<command>]]]]

sp_spaceusage 'display summary', {'table'|'index'}, <name>
              [,<where_clause> [,<order_by> [,<command>]]]

sp_spaceusage 'display', 'tranlog' [,{'syslogs'|NULL}
              [,<select_list> [,<where_clause> [,<order_by>]]]]

For more information, use:
sp_spaceusage 'help', 'display', 'table'
sp_spaceusage 'help', 'display', 'index'
sp_spaceusage 'help', 'display', 'tranlog'
```

Example 2 Displays a summary of the space usage on the titles table:

```
sp_spaceusage 'display summary', 'table', 'titles'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName    TableName    Type    UsedPages    RsvdPages    ExpRsvdPages
PctBloatRsvdPages
-----------  -----------  ------  -----------  ----------   --------------
----------
dbo          titles       DATA    6.0          30.0         16.0
87.50
dbo          titles       INDEX   8.0          64.0         32.0
50.00
```

Example 3 Displays the space usage information for the titles table:

```
sp_spaceusage 'display', 'table', 'titles'
```

```
All the page counts in the result set are in the unit 'KB'.
OwnerName    TableName    IndId    NumRows    UsedPages    RsvdPages
ExtentUtil     ExpRsvdPages    PctBloatUsePages     PctBloatRsvdPages
-----------  -----------  ------  -----------  ----------   --------------
----------     ------------    ----------------     -----------------
dbo          titles       0       18.0         6.0          30.0
20.00          16.0            0.0                  87.50
dbo          titles       1       NULL         4.0          32.0
12.50          16.0            0.00                 100.00
dbo          titles       2       NULL         4.0          32.0
12.50          16.0            0.00                 100.00
```

Example 4 Displays the space usage information, in megabytes, for all indexes on the titles table whose names start with title:

```
sp_spaceusage 'display using unit-MB', 'index', 'titles.title%'
```

```
All the page counts in the result set are in the unit 'MB'.
OwnerName    TableName    IndId     IndexName
UsedPages       RsvdPages
ExtentUtil      ExpRsvdPages       PctBloatUsedPages     PctBloatRsvdPages
----------   --------     ------     ------------  ----------      -------
----
------------    ----------------   ------------------    -----------------
---
dbo          titles       0          titles        .005859375      .029296
875
20.00           .015625           0.00                  87.50
dbo          titles       1          titleidind    .00390625       .03125
12.50           .015625           0.00                  100.00
dbo          titles       2          titleind      .00390625        .03125
12.50           .015625           0.00                  100.00
(1 row affected)
(return status = 0)
```

Example 5 Displays a summary of the space usage for all index names starting with *title* in the titles table:

```
sp_spaceusage 'display summary', 'index', 'titles.title%'

All the page counts in the result set are in the unit 'KB'.
OwnerName    TableName    IndexName    IndId    UsedPages
RsvdPages           ExpRsvdPages     PctBloatRsvdPages

--------    -----------  ----------   -------  --------------------
------------------  --------------  --------------
dbo         titles       titles       0         6.0
30.0                16.0            46.67
dbo         titles       titleidind   1         4.0
32.0                16.0            50.00
dbo         titles       titleind     2         4.0
32.0                16.0            50.00
```

Example 6 Displays a summary of the space usage for all indexes starting with *title* in the titles table where the value of PctBloatRsvdPages is less than 50:

```
sp_spaceusage 'display summary', 'index', 'titles.title%', 'where
PctBloatRsvdPages < 50'

All the page counts in the result set are in the unit 'KB'.
OwnerName    TableName    IndexName    IndId    UsedPages
 RsvdPages           ExpRsvdPages     PctBloatRsvdPages
--------    -----------  ----------   -------  --------------------
------------------  --------------  --------------
dbo         titles       titles       0         6.0
 30.0                16.0            46.67
```

Example 7 Displays a summary of the space usage for all indexes in the titles table in descending order of PctBloatRsvdPages where the value of PctBloatRsvdPages is greater than 30:

```
1> sp_spaceusage 'display summary', 'index', 'titles.title%', 'where
PctBloatRsvdPages > 30', 'order by PctBloatRsvdPages desc'

All the page counts in the result set are in the unit 'KB'.
OwnerName    TableName    IndexName    IndId    UsedPages
 RsvdPages           ExpRsvdPages     PctBloatRsvdPages

--------    -----------  ----------   -------  --------------------
 ------------------  --------------  --------------
dbo         titles       titleidind   1         4.0
 32.0                16.0            50.00
dbo         titles       titleind     2         4.0
 32.0                16.0            50.00
```

```
   dbo         titles       titles       0         6.0
    30.0                16.0            46.67
```

Example 8 Runs update table statistics on the authors table and summarizes its space usage information in the unit *pages*:

```
sp_spaceusage 'display summary using unit=pages', 'table', 'authors', null,
null, null, 'update table statistics'

All the page counts in the result set are in the unit 'pages'.
OwnerName TableName Type UsedPages RsvdPages ExpRsvdPages PctBloatRsvdPages
--------- --------- ---- --------- --------- ------------ -----------------
dbo        authors  DATA    2.0          16.0         8.0         100.00
dbo        authors  INDEX   4.0          32.0         16.0         50.00
```

Example 9 Displays the space usage information for the transaction log of the current database (pubs2):

```
sp_spaceusage 'display', 'tranlog'
TableName    TotalPages    UsedPages    CLRPagesFreePages
 PctUsedPages    PctFreePages
------------  ------------  -----------  ----------------------
 -------------  ------------
syslogs       4096.0        18.0 0.0     1482.0
 0.43            36.18
```

Example 10 Archives the space usage information for the authors table in the currrent database into the default table (spaceusage_object for tables and indexes):

```
sp_spaceusage 'archive', 'table', 'authors'
Data was successfully archived into table 'pubs2.dbo.spaceusage_object'.
```

Example 11 Archives the space usage information for the authors table into the default table (spaceusage_object for tables and indexes) in the pubs3 database, :

```
sp_spaceusage 'archive using dbname = pubs3', 'table', 'authors'
Data was successfully archived into table 'pubs3.dbo.spaceusage_object'.
```

Example 12 Runs update table statistics on the authors table and archives its space usage information into a table in the current database with the prefix monday_ (for this example, monday_spaceusage_object)

```
1> sp_spaceusage 'archive using dbname = pubs2, prefix=monday_',
'table','authors', null, 'update table statistics'
```

Example 13 Archives the space usage information for the transaction log of the current database into the default table (spaceusage_tranlog for transaction logs) in the pubs3 database, :

```
sp_spaceusage 'archive using dbname=pubs3', 'tranlog'
Data was successfully archived into table 'pubs3.dbo.spaceusage_tranlog'.
```

Example 14 Reports in detail the last archived space usage information for the authors table from the default table (spaceusage_object for table or index) in the current database:

```
sp_spaceusage 'report', 'table', 'authors'
All the page counts in the result set are in the unit 'KB'.
All the data in the result set are dated 'Jun 15 2007 11:50PM'.
OwnerName      TableName   IndId    NumRows    UsedPages    RsvdPages
ExtentUtil        ExpRsvdPages    PctBloatUsedPages    PctBloatRsvdPages
---------      ---------    -----   -------    ------       ---------
-------------     ------------    -----------------    --------------
dbo            authors      0       23.0        4.0        32.0
12.50             16.0            0.00                   100.00
dbo            authors      1       NULL        4.0        32.0
12.50             16.0            0.00                   100.00
dbo            authors      2       NULL        4.0        32.0
12.50             16.0            0.00                   100.00
(1 row affected)

(return status = 0)
```

Example15 Reports in summary the last archived space usage information for the authors table from the default table in the pubs3 database:

```
sp_spaceusage 'report summary using dbname=pubs3', 'table', 'authors'
All the page counts in the result set are in the unit 'KB'.
All the data in the result set are dated 'Jan 17 2007 11:29AM'.
OwnerName TableName Type UsedPages RsvdPages ExpRsvdPages PctBloatRsvdPages}
--------- --------- ---- ---------- -------- ------------ ------------------
dbo       authors    DATA      4.0      32.0         16.0              100.00
dbo       authors    INDEX     8.0      64.0         32.0               50.00
```

Example 16 Reports a summary from the monday_spaceusage_object table in the current database the last archived space usage information (in megabytes) for the authors table:

```
sp_spaceusage 'report summary using prefix=monday_, unit=MB', 'table',
'authors'
All the page counts in the result set are in the unit 'MB'.
All the data in the result set are dated 'Jan 17 2007 11:29AM'.
OwnerName   TableName   Type   UsedPages   RsvdPages   ExpRsvdPages
```

```
PctBloatRsvdPages
---------   ---------   ----   ---------   -------     ------------
-----------
dbo         authors     DATA   .00390625   .03125      .015625
100.00
dbo         authors     INDEX   .0078125    .0625        .03125
50.00
```

Example 17 Reports the space usage information from the default table in the current database for all the indexes on the authors table archived on Jun 9, 2007 or later:

```
sp_spaceusage 'report', 'index', 'authors.%', null, null, null, 'Jun 9 2007'
All the page counts in the result set are in the unit 'KB'.
ArchiveDateTime      OwnerName    TableName    IndId    IndexName   UsedPa
ges
RsvdPages    ExtentUtil   ExpRsvdPages    PctBloatUsedPages
PctBloatRsvdPages
-------------------  ---------    ---------    -----    ---------    -----
--
---------    ----------   ------------    -----------------
-----------------
Jun  9 2007 12:06AM   dbo           authors      0       authors      4.0
32.0         12.50        16.0             0.00
100.00
Jun 10 2007 12:05AM  dbo           authors      0       authors       4.0
32.0         12.50        16.0             0.00
100.00
Jun 11 2007 11:35PM  dbo           authors      0       authors       4.0
32.0         12.50        16.0             0.00
100.00
Jun  9 2007 12:06AM  dbo           authors      1       auidind       4.0
32.0         12.50        16.0             0.00
100.00
Jun 10 2007 12:05AM  dbo           authors      1       auidind       4.0
32.0         12.50        16.0             0.00
100.00
Jun 11 2007 11:35PM  dbo           authors      1       auidind       4.0
32.0         12.50        16.0             0.00
100.
Jun  9 2007 12:06AM  dbo           authors      2       aunmind       4.0
32.0         12.50        16.0             0.00
100.00
Jun 10 2007 12:05AM  dbo           authors      2       aunmind       4.0
32.0         12.50        16.0             0.00
100.00
Jun 11 2007 11:35PM  dbo           authors      2       aunmind       4.0
```

```
   32.0             12.50         16.0            0.00
   100.00
   (1 row affected)
   (return status = 0)
```

Example 18 Reports the space usage information for the authors table from the default table in the current database archived between Jun 10 2007 and Jun 15 2007:

```
sp_spaceusage 'report', 'table', 'authors', null, null, null, 'Jun 10 2007',
'Jun 15 2007'
All the page counts in the result set are in the unit 'KB'.
ArchiveDateTime      OwnerName   TableName   IndId   NumRows   UsedPages
RsvdPages   ExtentUtil   ExpRsvdPages   PctBloatUsedPages   PctBloatRsvdPages
------------------   --------    ---------    -----   -------   ---------
---------   ----------   ------------   -----------------   -----------------
Jun 10 2007 12:05AM  dbo         authors      0       23.0      4.0
32.0        12.50        16.0           0.00                100.00
Jun 11 2007 11:35PM  dbo         authors      0       23.0      4.0
32.0        12.50        16.0           0.00                100.00
Jun 13 2007 11:46PM  dbo         authors      0       23.0      4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      0       23.0      4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      0       23.0      4.0
32.0        12.50        16.0           0.00                100.00
Jun 10 2007 12:05AM  dbo         authors      1       NULL      4.0
32.0        12.50        16.0           0.00                100.00
Jun 11 2007 11:35PM  dbo         authors      1       NULL      4.0
32.0        12.50        16.0           0.00                100.00
Jun 13 2007 11:46PM  dbo         authors      1       NULL      4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      1       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      1       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 10 2007 12:05AM  dbo         authors      2       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 11 2007 11:35PM  dbo         authors      2       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 13 2007 11:46PM  dbo         authors      2       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      2       NULL     4.0
32.0        12.50        16.0           0.00                100.00
Jun 14 2007 11:46PM  dbo         authors      2       NULL     4.0
32.0        12.50        16.0           0.00                100.00
```

```
(1 row affected)
(return status = 0)
```

Usage

- sp_spaceusage provides space usage information for tables, indexes, and the transaction log of the current database.
- The set of columns that appear in the sp_spaceusage output depend on the action and entity type. By default, only a standard set of columns are displayed. However, you can include others with the *select_list* parameter, and you can view them all with the * wildcard in the select list.Table 20-1 and Table 20-2 provide the set of all output column names and their description for the entity types table, index and tranlog, respectively. Column names in the *select_list*, *where_clause*, *orderby_clause* parameters must belong to the set listed in these tables..

Table 20-1: Output columns for table or index entity types

Column name	Description
ArchiveDateTime	Timestamp of the data
ServerName	Server name
MaxPageSize	Logical page size, in *@@maxpagesize*
DBName	Object's database name
OwnerName	Object's owner name
TableName	Table name
Id	ID of the table
IndId	ID of the index
IndexName	Index name
PtnId	ID of the partition
PtnName	Partition name
DataPtnID	ID of the data partition whose data the index covers
RowSize	Number of rows in the partition
RowCount_ts	Number of rows in the partition as per the systabstats table
NumFwdRows	Number of forwarded rows in the partition
NumDelRows	Number of deleted rows in the partition
PctFwdRows	Percentage of rows that were forwarded in the partition
NonLeafRowSize	Average non-leaf row size in the partition
FF	Fill factor in the partition
MRPP	Maximum number of rows per page in the partition
ERS	Expected row size in the partition
RPG	Reserve page gap in the partition
IndexHeight	Height of the index tree in the partition
OAMAPageCount	Number of OAM and AP pages (in pages)

Column name	Description
Extent0PageCount	Number of pages in the extent 0 (in pages)
Status	status from sysindexes table
Sysstat	sysstat from sysobjects table
Sysstat2	sysstat2 from sysobjects table
LockScheme	Lock scheme of the table
NumVarCols	Number of variable columns the table has
HasAPLCI	Indicates whether the table has an APL CI
SpUtil	Space utilization derived statistic for the partition
DPCR	Data page cluster ratio derived statistic for the partition
DRCR	Data row cluster ratio derived statistic for the partition
IPCR	Index page cluster ratio derived statistic for the partition
LGIO	Large IO efficiency derived statistic for the partition
ExtentUtil	Extent utilization for the partition
EmptyPages	Number of empty pages in the partition (in units)
DataPages	Number of data pages in the partition (in units)
UsedPages	Number of used pages in the partition (in units)
RsvdPages	Number of pages reserved in the partition (in units)
LeafPages	Number of leaf pages in the partition (in units)
ExpDataPages	Expected number of data pages in the partition had the data been compact (in units)
ExpUsedPages	Expected number of used pages in the partition had the data been compact (in units)
ExpRsvdPages	Expected number of reserved pages in the partition had the data been compact (in units)
ExpLeafPages	Expected height of the index tree in the partition had the data been compact
PctBloatUsedPages	Percentage bloat in the used pages in the partition
PctBloatRsvdPages	Percentage bloat in the reserved pages in the partition
PctBloatLeafPages	Percentage bloat in the leaf pages in the partition
PctEmptyPages	Percentage of data pages that were empty in the partition

Table 20-2: Parameters available for the tranlog entity type

Column Name	Description
ArchiveDateTime	The timestamp of the data
ServerName	The server name
MaxPageSize	The logical page size set in *@@maxpagesize*
DBName	The object's database name
OwnerName	The object's owner name
TableName	The name of the transaction log, for example, syslogs
Id	ID of the syslogs table
IsMLD	Is it "mixed log and data" transaction log?

Column Name	Description
IsLogFull	Is the transaction log full?
LCTPages	The “last chance threshold” value of the log (in units)
TotalPages	Total number of log pages (in units)
UsedPages	The number of log pages already used (in units)
CLRPages	The number of log pages reserved for rollbacks (in units)
FreePages	The number of log pages that has not been used yet (in units)
PctUsedPages	The percentage of log pages that are in use already
PctFreePages	The percentage of log pages that are free

- The PctBloatUsedPages and PctBloatReservedPages columns give an estimate of how many more pages than the minimum the table is using and reserving, respectively. These values indicate how beneficial it may be for you to run reorg rebuild on the table.

  ExtentUtil is the ratio of the number of pages that are actually being used against the number of pages that are reserved for the object. Values closer to 100 indicate that most of the pages in the extents reserved for the object are currently used. Table 20-3 gives a synopsis of the measurements.

Table 20-3: Interpreting PctBloatUsedPages and PctBloatReservedPages values

PctBloatUsedPages value	PctBloatReservedPages value	Interpretation
Close to 0, low value	Close to 0, low value	Indicates the table is well compacted, and all allocated pages and allocation units are used completely. ExtentUtil should be close to 1.0
Close to 0, low value	Not close to 0, high value	Indicates the used pages are well compacted, but the table’s extents are under-utilized, and there is a large degree of interpage fragmentation, possibly due to large-scale deletions or empty pages. The unusedpgcnt in systabstats is probably also high. The high value of PctBloatReservedPages suggests that ExtentUtil to is probably much less than 1.0. You can probably resolve most issues by running reorg rebuild.
Not close to 0, high value	Close to 0, low value	Indicates a large degree of intrapage fragmentation, but a smaller degree of inter-page fragmentation. Because the extent utilization is probably high, ExtentUtil value should be close to 1.0. Running reorg compact will probably help resolve these issues.

PctBloatUsedPages value	PctBloatReservedPages value	Interpretation
Not close to 0, high value	Not close to 0, high value	A high value for PctBloatUsedPages indicates a large degree of intrapage fragmentation, where data rows in used pages are not fully compacted (the used pages contain most of the free space). Because interpage and intrapage fragmentation may cause he high value of PctBloatReservedPages, the value of Extent_Util may still be less than 1.0. Running reorg compact and reorg rebuild may resolve these issues.

- The database in which you are archiving the space usage data must have sp_dboption ... select into enabled.
- The archive tables are created if they do not already exist at the time of archiving, otherwise the results are appended to the current table. Because of this, any user running sp_spaceusage must have create table permission in the archive database.
- While archiving or reporting data, only tables owned by the user running sp_spaceusage are considered for the archive table. Tables with the same name but owned by another user are ignored. By default, the results are archived to or reported from the spaceusage_object table for tables or indexes and spaceusage_tranlog for the transaction log. .
- You can use the *from_date* and *to_date* arguments only for the report action when reporting from archived data. Adaptive Server uses only the data in the archive table that falls within the specified time-range when generating the report. If you do not include a *from_date* or a NULL, Adaptive Server uses all archived data prior to the *to_date*. If you do not include a *to_date* or NULL, Adaptive Server uses the current date as the value for *to_date*. If you do not include either the *from_date* or *to_date*, Adaptive Server uses the most recent data in the archive table to generate the report.
- sp_spaceusage results are estimated based on statistical data. These estimates are only as good the statistics provided. You can run update statistics to improve the accuracy of the results.

Permissions

Any user can run sp_spaceusage. However, they may not be able to view certain information about tables that they do not have persmissions to view.

Changed system procedures

sp_autoformat

sp_autoformat, which previously only supported columns of type char and varchar, accepts columns of datatypes int (smallint, bigint, tinyint, unsigned int), numeric, money, date/time, and float, real, and double precision.

sp_changedbowner

sp_changedbowner is used to change the owner of a database. You can now execute it with either sa_role or sso_role privileges. The owner of thresholds for that database is also changed to the specified user.

sp_checksource

sp_checksource encrypts the text of user-defined functions.

sp_configure

sp_configure now displays non-default value settings.

sp_configure [*configname* [, *configvalue*] | *group_name* | *non_unique_parameter_fragment*] [display_nondefault_settings]

sp_configure "configuration file", 0, {"write" | "read" | "verify" | "restore"} "*file_name*"

sp_depends

sp_depends checks for any object dependencies for user-defined functions.

sp_deviceattr

sp_deviceattar displays a warning message if the dsync option is disabled for a database device on file system.

sp_displaylogin

sp_displaylogin includes these changes:

- supports both a wildcard expression and a server user ID, and displays matching logins:

sp_displaylogin ['*user_id*' | '[loginame | *wildcard*]'

- *user_id* – user ID (suid) of the user whose login you are displaying.
- *wildcard* – wildcard character used for search purposes.

Displays the login account for the user with a suid of 56:

```
sp_displaylogin '56'
```

Displays the login account information for all users whose logins begin with "st":

```
sp_displaylogin 'st%'
```

- sp_displaylogin displays the encryption versions used for a login. For example, the last line of this output includes information about the login encryption Adaptive Server uses:

```
Suid: 1
Loginame: sa
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
    sa_role (default ON)
    sso_role (default ON)
    oper_role (default ON)
    sybase_ts_role (default ON)
Locked: NO
Date of Last Password Change: Mar  8 2007  3:04PM
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ANY
Login Password Encryption: SHA-256
```

If Adaptive Server uses encryption algorithms from earlier and the current release during a downgrade period, sp_displaylogin displays the earlier Sybase proprietary encryption algorithm and the new algorithm, SHA-256:

```
Login password encryption: SYB-PROP, SHA-256
```

When a login has been locked, sp_displaylogin shows the date, reason, and login that locked the account. For more information on sp_displaylogin, see Chapter 10, "Supported Security Features" for more information.

sp_droplogin

When sp_droplogin is unable to drop a login due to the existence – in any databasee – of a user in sysusers referencing the login suid, the names of databases in which the references are found are now displayed in the error message. The error message looks similar to:

```
1> sp_droplogin probe
2> go
Msg 19587, Level 16, State 1:
Procedure 'sp_droplogin', Line 281:
User exists or is an alias or is a database owner in
'master' 'sybsystemdb'  database(s).
(return status = 1)
```

sp_fixindex

sp_fixindex now works on a set of indexes rather than on a single index. sp_fixindex rebuilds the data layer if the target table has a placement or clustered index (it reclaims the unused space in the data layer while working on the placement or clustered index of a system table).

Syntax

This is the syntax for sp_fixindex (the new parameters are *index_name* and *force_option*):

sp_fixindex *database_name*, *table_name* [, *index_id* | null]
[, *index_name* | null] [, force_option]

Parameters

- *index_name* – indicates the index that needs to be processed. If a NULL value is used, the index associated with *index_id* is rebuilt. If *index_id* is also a NULL value, all the indexes in the system table are rebuilt
- force_option – forces Adaptive Server to rebuild the system table index in tempdb. sp_fixindex without the force_option forces the database specified by *database_name* to be in single-user mode, which is not possible for tempdb. Although the force_option allows you to rebuilt system catalogs in tempdb, it should not be used for user databases.

Examples

Example 1 Rebuilds the index with an index ID of 2 on testdb..sysprocedures:

```
sp_fixindex 'testdb', 'sysprocedures', 2
```

Example 2 Rebuilds the index csysprocedures in the testdb..sysprocedures system table:

```
sp_fixindex 'testdb', 'sysprocedures', null,
'csysprocedures'
```

Example 3 Rebuilds all available indexes on the sysprocedures table in testdb. If the table has clustered or placement index, sp_fixindex reclaims the unused space by removing the garbage present in data pages (that is, it rebuilds the data pages):

```
sp_fixindex 'testdb', 'sysprocedures'
```

Example 4 Rebuilds the index with an with an index ID of 2 on tempdb..sysprocedures:

```
sp_fixindex 'tempdb', 'sysprocedures', 2, null, 1
```

Example 5 Rebuilds the index csysprocedures for the table tempdb..sysprocedures:

```
sp_fixindex 'tempdb', 'sysprocedures', null,
'sysprocedures', 1
```

Example 6 Rebuilds all indexes on sysprocedures in tempdb:

```
sp_fixindex 'tempdb', 'sysprocedures', null, null, 1
```

sp_help

sp_help displays information about user-defined functions

sp_helpdevice

The description column of sp_helpdevice displays information about the device type. The device type is one of: `raw device`, `block device`, or `file system device`.

sp_helprotect

The new option, *permission_name*, in sp_helprotect provides information (grantor name, grantee name, table/column name, and grantability) for any specific permission granted in a given database. The syntax is:

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

permission_name can be any value from the sysprotect.action column.

sp_helptext

sp_helptext now reports the text of user-defined functions.

sp_helptext also has some new parameters, *numlines* and *printops*:

sp_helptext *objname* [,*grouping_num*] [, numlines [, printopts]]]

numlines

Specifies the numbers of lines for which to generate SQL text. The meaning of this argument can change, depending on the output properties you specify in .If printops includes showsql, numlines is treated as the number of lines of SQL text to display. If printops also includes context, numlines is the context block width that surrounds the starting line number. In that case, the SQL text generated is the output for the stated number of lines of context blocked around the requested starting line number.

printops

Supports comma-separated properties of the output format, selected by the user. You can specify one or more of these print options, in any order, as a comma-separated string:

- showsql – formatted SQL output for the compiled object. This argument passes control to sp_showtext; if showsql does not appear in the printopts list, the new functionality is not invoked.
- linenumbers – produces line numbers for each line of SQL output. There are no line numbers produced by default.
- comments – produces the line numbers as a comment field (*/; <nnn>/*), so that the SQL generated is usable for recreating the compiled object, if necessary, without further edits. No comment block is produced by default.
- context – produces a context block of output around the specified starting line numbers. If you supply no, or null, numlines parameter, a default context block of five lines is supplied. To get line numbers when displaying a context block of SQL, use `context, linenumbers`.
- noparams – suppresses the automatically generated parameter information. Use this print option to produce only the relevant portion of SQL output for the compiled object.
- ddlgen – generates the SQL text as DDL script, prefacing the output with a `use <database>` command, and a `drop <object>` command.

Note ddlgen and context are mutually exclusive print options. If you provide both, an error is raised.

sp_hidetext

sp_hidetext encrypts the text for user-defined functions.

sp_ldapadmin

Adaptive Server version 15.0.2 includes the following new syntax for sp_ldapadmin:

sp_ldapadmin 'set_max_ldapua_desc', max_ldapua_desc

sp_ldapadmin 'set_num_retries', num_retries

sp_ldapadmin 'set_log_interval', log_interval in minutes

Maximum LDAPUA descriptors per engine

The descriptor limit was previously hardcoded to 20 descriptors per engine. Now, you can set the descriptor limit using:

```
sp_ldapadmin 'set_max_ldapua_desc', max_ldapua_desc
```

Valid values for max_ldapua_desc are 1 to 20. The default value is 20.

Number of retries after transient errors

The number of LDAP API attempts was previously hardcoded to 3. Now, you can set the number of attempts using:

```
sp_ldapadmin 'set_num_retries', num_retries
```

Valid values for num_retries are 1 to 60. The default value is 3.

Error message and average time log interval

The log interval was previously hardcoded to 3 minutes. Now, you can set the log interval using:

```
sp_ldapadmin 'set_log_interval', log_interval
```

The log interval is specified in minutes. Valid values are 0 to 480 minutes. The default value is 3 minutes. 0 implies all messages are printed.

sp_locklogin

The "lock" option to sp_locklogin, when used with a value for "*number of inactive days*," locks inactive accounts that have not authenticated within that period. The following example locks all login accounts that have not authenticated within the past 60 days:

sp_locklogin 'all', 'lock', NULL, 60

The privileged role sso_role is required to lock logins.

This command has no effect if the sp_passwordpolicy option "enable last login updates" is set to "0".

The value for "*number of inactive days*" can be 1 to 32767 days.

The new syslogins columns lockdate, locksuid and lockreason are updated at time of locking/unlocking a login.

An administrator with sso_role can lock login accounts that have been inactive for a given number of days using the following sp_locklogin command:

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

See Chapter 10, “Supported Security Features” for more information about sp_locklogin.

sp_modifylogin account

Adaptive Server version 15.0.2 includes the following option for sp_modifylogin account:

```
sp_modifylogin account,  "max failed_logins", -1
```

This new value for the 'max failed_logins' option indicates that the failed login count in the syslogins column login count, is updated whenever an authentication failure occurs, but that the account is not locked. Compare with value 0, which avoids incrementing the column for every failed authentication and avoids locking the account due to authentication failures.

For more information on failed logins see “sp_configure” on page 269 for changes to maximum failed logins and “sp_passwordpolicy” on page 276 for changes to maximum failed login configuration options.

sp_modifystats

sp_modifystats allows the System Administrator, or any user with permission to execute the procedure and update statistics on the target table, to modify the density values of columns in sysstatistics.

sp_monitor

event options

Previously, sp_monitor event displayed all tasks (including system tasks), when called with no options. In versions 15.0.2 and above, the event option provides three possibilities:

- When no option is provided, only user tasks are displayed.
- When you specify sp_monitor, event, "-1", wait information about all tasks, both user and system, is displayed.
- When you specify sp_monitor, event, "spid", wait information pertaining to only the specified server process ID is displayed.

help option

The help option in sp_monitor also reports extensive information on using this procedure for deadlock analysis:

```
sp_monitor 'help', 'deadlock'
```

The help option also provides command-specific examples. For complete documentation of the help option see the *Reference Manual*.

sp_monitorconfig

Enhanced to create a table to hold the result set, if the user passes a table name for *result_tabl_name* that does not already exist.

sp_passwordpolicy

sp_passwordpolicy includes these changes:

- The set and clear commands in sp_passwordpolicy are now audited, through audit event 115, "Password Administration."
- Additional syntax:

  sp_passwordpolicy
  "enable last login updates",
  "allow password downgrade"
  "regenerate keypair",
  "expire login passwords", "[*login_name* | *wildcard*]"
  "expire role passwords", "[*role_name* | *wildcard*]"
  "expire stale login passwords", "datetime"
  "expire stale role passwords", "datetime"
  "maximum failed logins", -1

  Where:

  - *login_name* – is the name of the login whose password you are expiring.
  - *role_name* – is the name of role whose password you are expiring.
  - *wildcard* – is a wildcard character used for search purposes.
  - enable last login updates – is used to enable or disable code in Adaptive Server authentication that records the timestamp when each login occurs.The first parameter "set" sets the value of this attribute.The parameter "list" displays the current value of the attribute, and the parameter "clear" deletes the row from sysattributes. On upgrade or new installation, this attribute does not exist in sysattributes. The login timestamp occurs when the attribute row does not exist or has a value of 1. The login timestamp is not maintained if the attribute value is 0:

sp_passwordpolicy "set", "enable last login updates", 1 or 0

sp_passwordpolicy "list", "enable last login updates"

sp_passwordpolicy "clear", "enable last login updates"

- allow password downgrade – ends the password downgrade period. During the password downgrade period, passwords are stored in syslogins in both old and new encodings to allow user passwords to retained if the server is downgraded, for example, to Adaptive Server 15.0.2:

  sp_passwordpolicy 'set' 'allow password downgrade', 0

- regenerate keypair – generates the asymmetric key pairs to be used for network login password encryption.There is no catalog update for this option; the actions occur only in memory fields.
- expire state login passwords – login passwords expire on the specified date.
- expire state role passwords – role passwords expire on the specified date.
- *date_time* – expiration date.
- "maximum failed logins", -1 – This new value for maximum failed logins indicates that the failed login count in the syslogins column login count is updated whenever an authentication failure occurs, but that the account is not locked. Compare with a 0 value, which avoids incrementing the column for every failed authentication and avoids locking the account due to authentication failures. See "sp_configure" on page 269 and "sp_modifylogin account" on page 275 for additional changes to maximum failed login configuraiton.

For example, this expires all logins that begin with "bob" on March 2, 2007:

```
sp_passwordpolicy 'expire login passwords' 'bob%'
```

In the following example, roles with passwords that have not been changed after February 2, 2006 are expired:

```
sp_passwordpolicy 'expire role passwords', '020208'
```

sp_sendmsg

In previous releases, the maximum length for a message sent with this system procedure was 255 characters. For Adaptive Server release 15.0.2, the maximum length of a sp_sendmsg message is 4096 characters.

sp_tempdb

Two options of sp_tempdb are enhanced:

```
sp_tempdb show,'db'
sp_tempdb who,'db_name'
```

sp_tempdb show, 'db'

This option displays all temporary databases and the names of the groups to which the temporary databases belong. The following SQL commands illustrate this:

```
create temporary database mytempdb
-------------
CREATE DATABASE: allocating 1536 logical pages (3.0
megabytes) on disk 'master'.

create temporary database mytempdb1
----------
CREATE DATABASE: allocating 1536 logical pages (3.0
megabytes) on disk 'master'.

sp_tempdb 'add', mytempdb,'default'
------------
(return status = 0

sp_tempdb show, db
-------------
Database Group
--------
tempdb default
mytempdb default
mytempdb1
(3 rows affected)
(return status = 0)
```

sp_tempdb who, 'db'

This option displays the login and application names of all active sessions assigned to specified temporary databases.

```
sp_addlogin anunay, anunay
---------------
sp_tempdb "bind", lg, sa, DB, mytempdb3
-------------
(return status = 0)

sp_tempdb "bind", lg, anunay, DB, mytempdb3
-----------------
(return status = 0)
```

```
starting sessions
--------------------
${ISQL} -J -U anunay -P anunay -I${SYBASE}/interfaces -w200

sp_tempdb who, mytempdb3
-------------
spid loginame application
------ -------- -----------
11 sa isql
13 anunay isql

(2 rows affected)
(return status = 0)
```

sp_who

A new column, tempdbname, displays temporary database names of all active sessions.

```
sp_who
 fid spid status loginame origname hostname blk_spid dbname tempdbname
           cmd         block_xloid
 --- ---- -------- -------- -------- -------- -------- ------ ----------
       ----------------- -----------
   0    2 sleeping NULL   NULL     NULL            0 master tempdb
            DEADLOCK TUNE        0
   0    3 sleeping NULL   NULL     NULL            0 master tempdb
            SHUTDOWN HANDLER     0
   0    4 sleeping NULL   NULL     NULL            0 master tempdb
            ASTC HANDLER         0
   0    5 sleeping NULL   NULL     NULL            0 master tempdb
            CHECKPOINT SLEEP     0
```

New and changed system tables

New system tables

sysoptions

sysoptions is a fake table queried by sp_options. The columns names are case sensitive.

Table 20-4: sysoptions

Name	Datatype	Attributes	Description
spid	int		Contains the process ID.
name	varchar(100)		Contains the name of the option.
category	varchar(100)		Contains the name of the category to which the option belongs.
currentsetting	varchar(100)	NULL	Contains the current setting of the option.
defaultsetting	varchar(100)	NULL	Contains the default setting of the option.
scope	int		Contains the bitmap used to capture information about options. The bits are ordered as follows: • Bit 1 – compiled time options • Bit 2 – stored procedure specific options • Bit 3 – binary options

Changed system tables

sysquerymetrics

Adaptive Server version 15.0.2 increases the number of metrics shared among user IDs. The change reduces the number of entries in sysquerymetrics (a view of sysqueryplans), and automatically aggregates the metrics for identical queries across different user IDs.

The user ID (uid) of sysquerymetrics is 0 when all table names in a query that are not qualified by user name are owned by the DBO.

For example, if table t1 is owned only by the DBO and shared by different users:

```
select * from t1 where c1 = 1
```

Adaptive Server uses 0 as the uid for the sysquerymetrics entry for all users executing this query who do not have a private table named t1.

In this example, if table t2 is owned and qualified by "user1," Adaptive Server also uses an UID of 0:

```
selet * from user1.t2 where c1 = 1
```

However, if table t3 is owned only by "user1," but is unqualified and not owned by the DBO, the UID of "user1" is used in the sysquerymetrics entry:

```
select * from t3 where c1 = 1
```

syscolumns

Adds a status bit to the status2 column that indicates an encrypted column has a decrypt_default attached to it:

0x00001000 (4096) – column has a decrypt default

sysobjects

The type column of sysobjects includes an entry of "DD" for each object that has a decrypt default.

sysaudits

Changes include:

- The Alter Encryption Key audit event name is changed to AEK As/Not Default
- Adaptive Server release 15.0.2 supports these audit events and numbers:
  - 118 – AEK Modify Encryption
  - 119 – AEK Add Encryption
  - 120 – AEK Drop Encryption
  - 121 – AEK Modify Owner
  - 122 – AEK Key Recovery

sysattributes

sysattributes includes these changes:

- Adds two classes:
  - Class 31 allow password downgrade – when set to 1, allow password downgrade enables special handling of login passwords for compatibility with Adaptive Server release 15.0 and earlier.
  - Class 32 enable last login updates – when set to 1, enable last login updates enables system tables to store the date of the last login.
- sysattributes includes information about default decrypt. These are the changes to the columns:
  - attribute – specifies a default decrypt on an encrypted column with a value of 1 (DECRYPT-DEFAULT_ID) for objects with a type of EC and a class of 25.
  - object – includes the decypt default ID.
  - object_info_1 – includes the table ID for a table whose encrypted column defines the decrypt default.
  - object_info2 – specifies the *colid* of the encrypted column that includes the decrypt default.

sysencryptkeys

Changes to sysencryptkeys includes

- New types:
  - EK_KEYCOPY – 0x0010,
  - EK_KEYBASE – 0x0020
  - EK_RECOVERY – 0x0040
- New status bits:
  - EK_KEYRECOVERY(0x00000004) – keys encrypted for lost password protection.
  - EK_LOGINACCESS(0x00000008) – key encrypted for login access
  - EK_LOGINPASS (0x00000010) – key encrypted with login password
  - EK_USERPWD(0x00000100) – keys encrypted with user-encryption passwords
- Changes to the description for the uid column – user access or key recovery row. uid contains the user ID (uid) identifyinig the database user associated with current row.

Previous versions of Adaptive Server did not use the uid column.

syslogins

Included these changes:

- In previous releases the maximum length of the password column was 30 bytes. In Adaptive Server release 15.0.2, the maximum length of the password column is 128 bytes.
- Adds these columns:

Name	Datatype	Description
lastlogindate	datetime	Timestamp for the user's last login.
crdate	datetime	Timestamp when the login was created.
locksuid	int	The server user ID (suid) responsible for locking the login.
lockreason	int	Reasons for lock; one of: • NULL – account has not been locked • 0 – locksuid locked the account by executing sp_locklogin • 1 – locksuid locked the inactive account by executing sp_locklogin 'all', 'lock', 'ndays' • 2 – Adaptive Server locked the account because the number of failed login attempts reached max failed logins. • 3 – locksuid locked the account because the password downgrade period has ended and a login or role was not transitioned to SHA-256
lockdate	datetime	Timestamp when the login was locked.

syssrvroles

In previous releases the maximum length of the password column was 30 bytes. In Adaptive Server release 15.0.2, the maximum length of the password column is 128 bytes.

New and changed configuration parameters

New configuration parameters

Configuration parameters are user-definable settings that control various aspects of Adaptive Server behavior. Adaptive Server supplies default settings for all configuration parameters. You can use configuration parameters to tailor Adaptive Server for an installation's particular needs. You can find a complete list of configuration parameters in the *System Administration Guide for Adaptive Server*.

enable merge join

Summary information	
Default value	2
Range of values	0 – 2
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

enable merge join enables or disables merge join at the server level. Setting enable merge join to:

- 0 – disables merge joins at the server level
- 1 – enables merge joins at the server level.
- 2 – sets merge joins to their default values at the server level

The default value for merge join depends on current value of the optimization goal configuration parameter:

Value for optimization goal	Default value for merge join
allrows_mix	on
allrows_dss	on
allrows_oltp	off
fastfirstrow	off

cost of a logical io

Summary information	
Default value	2
Range of values	0 - 254
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

cost of a logical io specifies the cost of a single logical IO.

cost of a physical io

Summary information	
Default value	25
Range of values	0 - 254
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

cost of a phsyical io specifies the cost of a single physical IO.

cost of a cpu unit

Summary information	
Default value	1000
Range of values	0 - 65534
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

cost of a cpu unit specifies the cost of a single CPU operation.

enable encrypted columns

Summary information	
Default value	0
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

Set enable encrypted columns to 1 (on) to use the encryption functionality.The option is turned off in a server that has the liscense to use encrypted columns. any commands against encrypted columns fail with an error. You must set both the configuration parameter and the license option to enable encryption.

```
sp_configure 'enable encrypted columns', 1
go
```

Adaptive Server 15.0.2 changes enable encrypted columns from static to dynamic. In other words, you do not need to restart Adaptive Server for the parameter to take effect.

max online Q engines

Summary information	
Default value	0
Range of values	0–127
Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	

max online Q engines required for MQ series and specifies the maximum number of Q engines you can have online. You may need to increase max online engines to accomodate the number of max online Q engines.

Add 2 more to max online Q engines assuming max online Q engines is at 4.

```
sp_configure 'max online engines' 6
go
sp_configure 'max online Q engines'
go
```

metrics elap max

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

metrics elap max has an effect only when enable metrics capture is on. If the elapsed time of the query is less than the value of this configuration parameter, then the query metrics associated with this query are not written to the system tables, avoiding excessive catalog writes for simple queries.

metrics exec max

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

metrics exec max has an effect only when enable metrics capture is on. If the execution time of the query is less than the value of this configuration parameter, then the query metrics associated with this query are not written to the system tables, avoiding excessive catalog writes for simple queries.

metrics lio max

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

metrics lio max has an effect only when enable metrics capture is on. If the logical IO time of the query is less than the value of this configuration parameter, then the query metrics associated with this query are not written to the system tables, avoiding excessive catalog writes for simple queries.

metrics pio max

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

metrics pio max has an effect only when enable metrics capture is on. If the physical IO time of the query is less than the value of this configuration parameter, then the query metrics associated with this query are not written to the system tables, avoiding excessive catalog writes for simple queries.

min pages for parallel scan

Summary information	
Default value	200
Range of values	0–2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

parallel scan lets you run queries in parallel. The min pages for parallel scan parameter is a lower to avoid having small queries produce overhead during parallel processing. There is no benefit in having small queries processed in parallel.

net password encryption reqd

Summary information	
Default value	0
Range of values	0–2
Status	Dynamic
Display level	Comprehensive
Required role	System Security Officer
Configuration group	Security

The administrator sets this configuration parameter to require all incoming login authentication requests using Adaptive Server authentication to encrypt the login password when transmitted on the network. Network password encryption is used to keep passwords secure on the network.

number of Q engines at startup

Summary information	
Default value	0
Range of values	0–127
Status	Static
Display level	Comprehensive
Required role	System Administrator
Configuration group	

number of Q engines at startup required for MQ series and specifies the maximum number of Q engines you can have online. You may need to increase max online engines to accomodate the number of Q engines at startup.

This example shows adding 2 to ‘max on'ine engines’ assuming curent ‘max online engines’ is 4.

```
sp_configure 'number of Q engines at startup', 2
go
```

prod-consumer overlap factor

Summary information	
Default value	20
Range of values	

Summary information	
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

prod-consumer overlap factor affects optimization. Adaptive Server changes the group by algorithm, and you cannot use set statistics I/O with parallel plans.

send doneinproc tokens

Summary information	
Default value	1
Range of values	0–1
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	

send doneinproc tokens replaces the dbcc tune option doneinproc the trace flag 292. Any queries that are in process are affected by any changes to this parameter.

Change the default value of this option with care. doneinproc tokens are crucial to client behavior in CT-library and DB-library. Using the value 0 may cause state-machine errors in some CT-Library applications.

Changed configuration parameters

max async i/os per engine

Note The traceflags 1630, 1639 and 1649 have been rendered obsolete by the KAIO implementation. Support for these traceflags has been removed from Adaptive Server.

In earlier versions of Adaptive Server on Linux, the default for max async i/os per engine was 2147483647. In Adaptive Server version 15.0.2, this default has changed to 1024.

On the Linux platform, max async i/os per engine controls the number of asynchronous IOs each engine reserves from the operating system when the machine starts. Your system may benefit from using a number greater than the default value.

You can use sp_sysmon to help tune the max async i/os per engine parameter. sp_sysmon's `disk i/o section` contains information about the maximum number of outstanding IOs for each engine during the sample period and the number of IOs that were delayed because of engine or operating system limits. Generally, any IOs delayed by engine limits indicate that you should increase the value of max async i/os per engine.

Whether Adaptive Server can perform asynchronous IO on a device depends on whether or not this device support KAIO. The Linux kernel requires that you implement KAIO support at the filesystem level. Most major filesystems provide support for KAIO, including ext3, xfs, jfs, and raw devices (The tmpfs file system does not support KAIO). If the device does not support KAIO, Adaptive Server cannot perform asyncronous IO on that device, and instead reverts to standard synchronous IO for all reads and writes to that device. Adaptive Server prints a message similar to the following in the errorlog indicating that the device has switched to synchronous IO:

```
00:00000:00001:2006/12/15 11:47:17.98 kernel  Virtual device
'/dev/shm/tempdb.dat' does not support kernel asynchronous i/o. Synchronous i/o
will be used for this device.
```

Maximum failed logins

The new -1 value for maximum failed logins indicates that the failed login count in the syslogins column logincount is updated whenever an authentication failure occurs, but that the account is not locked. Compare with a o value, which avoids incrementing the column for every failed authentication and avoids locking the account due to authentication failures.

See "sp_passwordpolicy" on page 276 and "sp_modifylogin account" on page 275 for more information.

Print deadlock information

print deadlock information adds a new parameter of 2, which allows you to print a summary of deadlock information to the errorlog. For example:

```
Deadlock Id 34: Process (Familyid 0, Spid 70) was waiting for a 'exclusive page'
lock on page 10858346 of the 'equineline_job' table in database 18 but process
(Familyid 0, Spid 88) already held a 'exclusive page' lock on it.
Deadlock Id 34: Process (Familyid 0, Spid 88) was waiting for a 'exclusive page'
lock on page 11540986 of the 'equineline_job' table in database 18 but process
(Familyid 0, Spid 70) already held a 'update page' lock on it.
```

New and changed functions

Changed functions

used_pages

In previous versions, Adaptive Server treated a value of either `indid=0` or `indid=1`, passed to used_pages(), as identical, and in both cases returned the used pages at the data layer as well as the clustered index layer.

In Adaptive Server version 15.0.2, in an all-pages-locked table with a clustered index, used_pages() is now passed only the used pages in the data layer, for a value of `indid = 0`. When `indid=1` is passed, the used pages at the data layer and at the clustered index layer are returned, as in previous versions.

New functions

authmech()

This function returns the authentication method used for a logged-in server process ID session.

authmech ([spid)]

hashbytes

Description

Produces a fixed-length, hash value expression.

Syntax

hashbytes(*algorithm*, *expression*[, *expression...*] [, using *options*])

Parameters

- *expression*[, *expression*...] – is the value to be hashed. This value can be a column name, variable, constant expression, or a combination of these that produces a single value. It cannot be image, text, unitext, or off-row Java datatypes.
- *algorithm* – is the algorithm used to produce the hash value. A character literal (not a variable or a column name) that can take the values "md5", "sha", "sha1", or "ptn".
  - Md5 (Message Digest Algorithm 5) – is the cryptographic hash algorithm with a 128 bit hash value.
    - hashbytes('md5', *expression*[,...]) results in a varbinary 16-byte value.
  - Sha-Sha1 (Secure Hash Algorithm) – is the cryptographic hash algorithm with a 160-bit hash value.
    - hashbytes('sha1', *expression*[,...]) results in a varbinary 20-byte value.
  - Ptn – is the partition hash algorithm with 32-bit hash value. The *using* clause is ignored for the 'ptn' algorithm.
    - hashbytes(*'ptn'*, *expression*[,...]) results in an unsigned int 4-byte value.
  - using – orders bytes for platform independence. The optional using clause can precede the following option strings:
    - lsb – all byte-order dependent data is normalized to little-endian byte-order before being hashed.
    - msb – all byte-order dependent data is normalized to big-endian byte-order before being hashed.
    - unicode – character data is normalized to unicode (UTF–16) before being hashed.

      Note A UTF – 16 string is similar to an array of short integers. Because it is byte-order dependent, Sybase suggest for platform independence you use lsb or msb in conjunction with UNICODE.

    - unicode_lsb – a combination of unicode and lsb.
    - unicode_msb – a combination of unicode and msb.

Examples

This update is used to seal each row of a table against tampering. This example assumes the existence of a user table called "xtable" and col1, col2, col3 and tamper_seal.

```
update xtable set tamper_seal=hashbytes('sha1', col1,
col2, col4, @salt)
--
declare @nparts unsigned int
select @nparts= 5
select hashbytes('ptn', col1, col2, col3) % nparts from
xtable
```

This example shows how col1, col2, and col3 will be used to partition rows into five partitions.

```
alter table xtable partition by hash(col1, col2, col3) 5
```

Usage

The algorithm parameter is not case-sensitive; "md5," "Md5" and "MD5" are all equivalent. However, if the *expression* is specified as a character datatype, the value is case sensitive. "Time," "TIME," and "time" will produce different hash values.

Note Trailing null values are trimmed by Adaptive Server when inserting into varbinary columns.

In the absence of a using clause, the bytes that form *expression* are fed into the hash algorithm in the order they appear in memory. For many datatypes, order is significant. For example, the binary representation of the 4-byte INT value 1 will be 0x00, 0x00, 0x00, 0x01, on MSB-first (big-endian) platforms and 0x01, 0x00, 0x00, 0x00 on LSB-first (little-endian) platforms. Because the stream of bytes is different for different platforms, the hash value is different as well.

With the using clause, the bytes that form *expression* can be fed into the hashing algorithm in a platform-independent manner. The using clause can also be used to transform character data into Unicode so that the hash value becomes independent of the server's character configuration.

Note The hash algorithms MD5 and SHA1 are no longer considered entirely secure by the cryptographic community. Be aware of the risks of using MD5 or SHA1 in a security-critical context.

Standards

SQL92- and SQL99-compliant

Permissions

Any user can execute hashbyte.

See also

See also hash for platform dependent hash values.

hash

Description

Produces a fixed-length hash value expression.

Syntax

hash(*expression* , [*algorithm*])

Parameters

- *expression* – value to be hashed. This can be a column name, variable, constant expression, or any combination of these that evaluates to a single value. It cannot be image, text, unitext, or off-row Java datatypes. Expression is usually a column name. If expression is a character constant, it must be enclosed in quotes.
- *algorithm* – the algorithm used to produce the hash value. A character literal (not a variable or column name) that can take the values of either "md5" or "sha1", 2 (meaning "md5" binary), or 3 (meaning "sha1" binary). If omitted "md5" is used.
  - hash(*expression*, 'md5') results in a varchar 32-byte string
    - Md5 (Message Digest Algorithm 5) – is the cryptographic hash function with a 128- bit hash value.
  - hash(*expression*) – results in a varchar 32-byte string
  - hash(*expression*, *'sha1'*) – results in a varchar 40-byte string
    - Sha1 (Secure Hash Algorithm) – is the cryptographic hash function with a 160-bit hash value.
  - hash(*expression*, *2*) – results in a varbinary 16-byte value (using the 'md5' algorithm)
  - hash(*expression*, *3*) – results in a varbinary 20-byte value (using the 'sha1' algorithm)

Examples

This example shows how a seal is implemented. The existence of a table called "atable" and with columns id, sensitive_field and tamper seal.

```
update atable set tamper_seal=hash(convert(varchar(30),
id) + sensitive_field+@salt, 'sha1')
```

Usage

When specified as a character literal, *algorithm* is not case-sensitive—"md5", "Md5" and "MD5" are equivalent. However, if *expression* is specified as a character datatype then the value is case sensitive. "Time," "TIME," and "time" will produce different hash values.

If *algorithm* is a character literal, the result is a varchar string. For "md5" this is a 32-byte string containing the hexadecimal representation of the 128-bit result of the hash calculation. For "sha1" this is a 40-byte string containing the hexadecimal representation of the 160-bit result of the hash calculation.

If *algorithm* is an integer literal, the result is a varbinary value. For 2, this is a 16-byte value containing the 128-bit result of the hash calculation. For 3, this is a 20-byte value containing the 160-bit result of the hash calculation.

Note Trailing null values are trimmed by Adaptive Server when inserted into varbinary columns.

Individual bytes that form *expression* are fed into the hash algorithm in the order they appear in memory. For many datatypes order is significant. For example, the binary representation of the 4-byte INT value 1 will be 0x00, 0x00, 0x00, 0x01 on MSB-first (big-endian) platforms and 0x01, 0x00, 0x00, 0x00 on LSB-first (little-endian) platforms. Because the stream of bytes is different between platforms, the hash value is different as well. Use hashbytes function to achieve platform independent hash value.

Note The hash algorithms MD5 and SHA1 are no longer considered entirely secure by the cryptographic community. As for any such algorithm, you should be aware of the risks of using MD5 or SHA1 in a security-critical context.

Standards — SQL92- and SQL99- compliant

Permissions — Any user can execute hash.

See also — See also hashbytes for platform independent hash values.

index_name()

index_name() returns the name of an index, when you specify the ID of the index and the database, and the object on which the index is defined.

asehostname

Description — Returns the physical or virtual host on which Adaptive Server is running.

Syntax

```
asehostname
```

Parameters — asehostname does not require any parameters.

Examples

```
select asehostname()
```

```
-----------------------------------
linuxkernel.sybase.com
```

Usage

Standards

SQL/92 and SQL/99 compliant

Permissions

Only users with the sa_role can execute asehostname.

reserve_identity

Description

reserve_identity allows a process to reserve a block of identity values for use by that process.

After a process calls reserve_identity to reserve the block of values, subsequent identity values needed by this process are drawn from this reserved pool. When these reserved numbers are exhausted, or if you insert data into a different table, the existing identity options apply. reserve_identity can retain more than one block of identity values, so if inserts to different tables are interleaved by a single process, the next value in a table's reserved block is used.

Reserves a specified size block of identity values for the specified table, which are used exclusively by the calling process. Returns the reserved starting number, and subsequent inserts into the specified table by this process use these values. When the process terminates, any unused values are eliminated.

Syntax

reserve_identity (*table_name*, *number_of_values*)

Parameters

- *table_name* – name of the table for which the reservation are made. The name can be fully qualified; that is, it can include the *database_name*, *owner_name*, and *object_name* (in quotes).
- *number_of_values* – number of sequential identity values reserved for this process. This must be a positive value that will not cause any of the reserved values to exceed the maximum values for the datatype of the identity column.

Examples

This example describes a typical usage scenario for reserve_identity, and assumes that table1 includes col1 (with a datatype of int) and a col2 (an identity column with a datatype of int). This process is for spid 3:

```
select reserve_identity( table1, 5 )
---------------
10
```

Insert values for spids 3 and 4:

```
Insert table1 values(56) -> spid 3
Insert table1 values(48) -> spid 3
```

```
Insert table1 values(96) -> spid 3
Insert table1 values(02) -> spid 4
Insert table1 values(84) -> spid 3
```

Select from table table1:

```
select * from table1
Col1          col2
--------       -----
3              1-> spid 3 reserved 1-5
3              2-> spid 3
3              3-> spid 3
4              6<= spid 4 gets next unreserved value
3              4<= spid 3 continues with reservation
```

The result set shows that spid 3 reservered identity values 1 – 5, spid 4 receives the next unreserved value, and then spid 3 reserves the subsequent identity values.

Usage

- The return value, *start_value*, is the starting value for the block of reserved identity values. The calling process uses this value for the next insert into the specified table
- reserve_identity allows a process to:
  - Reserve identity values without issuing an insert statement.
  - Know the values reserved prior issuing the insert statement
  - “Grab” different size blocks of identity values, according to need.
  - Better control “over gaps” by reserving only what is needed (that is, they are not restricted by preset server grab size
- Values are automatically used with no change to the insert syntax.
- NULL values are returned if:
  - A negative value or zero is specified as the block size.
  - The table does not exist.
  - The table does not contain an identity column.
- If you issue reserve_identity on a table in which this process has already reserved these identity values, the function succeeds and the most recent group of values is used.

- You cannot use reserve_identity to reserve identity values on a proxy table. Local servers can use reserve_identity on a remote table if the local server calls a remote procedure that calls reserve_identity. Because these reserved values are stored on the remote server but in the session belonging to the local server, subsequent inserts to the remote table use the reserved values.
- If the identity_gap is less than the reserved block size, the reservation succeeds by reserving the specified block size (not an identity_gap size) of values. If these values are not used by the process, this results in potential gaps of up to the specified block size regardless of the identity_gap setting.

Permissions

You must have insert permission to reserve identity values.

New and changed global variables

New global variables

@@lastlogindate

Global T-SQL variable *@@lastlogindate* is available to each user login session. A datetime datatype, its value is the lastlogindate column for the login account before the current session was established. This variable is specific to each login session and can be used by that session to determine the previous login to the account. If the account has not been used previously or "sp_passwordpolicy 'set', enable last login updates" is 0, then the value of *@@lastlogindate* is NULL.

Changed global variables

@@opttimeoutlimit

Previous version of Adaptive Server user documenation listed *@@opttimeout* as a server global variable that displays the current optimization timeout limit for query optimization.

This is incorrect. The actual name of the global variable that displays the current optimization timeout limit for query optimization is *@@opttimeoutlimit*

New trace flags

- 15340 enables server wide, no matter advanced_aggregation
- 15341 disables server wide, no matter advanced_aggregation

PART 2

New features for Adaptive Server Version 15.0.1

Part 2 includes descriptions for these features:

CHAPTER 21

New Features in Adaptive Server 15.0.1

This describes features introduced with Adaptive Server version 15.0.1.

Changes to *sp_sysmon*

Adaptive Server version 15.0.1 changes the default behavior of sp_sysmon.

In Adaptive Server version 15.0.1 and later, sp_sysmon does not clear the monitor counters. You no longer have to specify the noclear option to prevent sp_sysmon from clearing the monitor counters. This is the default behavior.

If you need to clear the monitor counters, use sp_sysmon with the clear option. For compatibility reasons, Adaptive Server accepts the noclear option as a valid parameter, but it does not affect the behavior of sp_sysmon.

However, if you run sp_sysmon using the begin_sample and end_sample options to begin and end the sample period, sp_sysmon always clears the monitor counters. Adaptive Server issues an error message if you run sp_sysmon with begin_sample or end_sample and the noclear option.

Note For information about the noclear option that was introduced in Adaptive Server version 12.5.3, see Chapter 43, “Monitor Counters and sp_sysmon.”

Changes to abstract plans

The abstract plan syntax, used by Adaptive Server Enterprise to force the query plan chosen by the optimizer, has been extended to allow several query-level settings, that were previously available only at the session level.

The optimization criteria are handled at the session level by the following set statements:

```
set
        nl_join|merge_join|hash_join|...
        0|1
```

The use ... abstract plan syntax has been extended to accept any number of use forms before the abstract plan derived table. Previously, optgoal and opttimeout could not be in the same abstract plan with a derived table. For example, this statement would need to be separate from a optgoal statement in a query:

```
select ...
       plan
       "(use opttimeoutlimit 10) (i_scan r)"
```

However, with Adaptive Server 15.0.1, you can include several statements in the same abstract plan in two ways:

- By using several use statements, for example:

```
select ...
plan
"(use optgoal allrows_dss) (use nl_join off) (...)"
```

- By placing several items within one use form, for example:

```
select ...
    plan
 "(use (optgoal allrows_dss) (nl_join off)) (...)"
```

At the query level, use the optimization goal (opt_goal) or timeout (opttimeout) setting with the use ... abstract plan syntax. At the session level, use these settings with the set plan ... syntax:

- Optimization goal.
- Optimization timeout.

For example, join r outer to s and enable the hash_join without an optimization goal (opt_goal):

```
select ...
>
>          plan
```

```
>
>              "(use hash_join on)
>
>              (join (scan r) (scan s))"
```

This example uses the opt_goal and allrows_oltp statements, but with hash_join enabled:

```
select ...
>
>     plan
>
>         "(use opt_goal allrows_oltp)(use hash_join
on)"
```

When setting the optimization goal and the optimization criteria at the query level, the order of the use statements does not affect the outcome.

- The abstract plan optimization goal is set first, and sets the optimization goal defaults for the optimization criteria.
- Abstract plan optimization, which supersedes optimization goal defaults criteria, can be set after you set the optimization goal.

Operator name alignment for abstract plan and optimizer criteria

The names of algorithms differ between their usage in abstract plans and in the set command. For example, a hash join is called h_join in abstract plans, but is called hash_join in the set command. Adaptive Server accepts both keywords in the extended abstract plan syntax. For example:

```
select ...

plan

"(h_join (t_scan r) (t_scan s))"
```

is equivalent to:

```
 select ...

   plan

     "(hash_join (t_scan r) (t_scan s))"
```

and:

```
select ...
```

```
plan
"(use h_join on)"
```

and:

```
select ...
plan
"(use hash_join on)"
```

When a table abstract plan is present, it takes precedence:

```
select ..
from r, s, t
...
plan
"(use hash_join off)
(h_join (t_scan r) (t_scan s))"
```

The query uses the hash_join for **r** and **s** scans; but for the join with **t** it does not use hash_join as specified by the use abstract plan form, since it was not specified in the table abstract plan.

Extending the optimizer criteria *set* syntax

The set <*opt criteria*> statement, with a 0 or 1 syntax, accepts *on/off/default*, where *default* indicates that you are using the current optimization goal setting for this optimization criteria (for the complete syntax for set, see *Reference Manual: Commands*).

Literal parameterization

In versions of Adaptive Server earlier than 15.0.1, two queries that were identical except for one or more literal values resulted in the statement cache storing two separate query plans, or two additional rows in sysqueryplans. For example, the query plans for these queries were stored separately, even though they are almost identical:

```
select count(*) from titles where total_sales > 100
select count(*) from titles where total_sales > 200
```

Adaptive Server version 15.0.1 allows you to automatically convert literal values in SQL queries to parameter descriptions (similar to variables). An sp_configure option supports this feature, which is called enable literal autoparam.

To enable or disable enable literal autoparam server-wide, use:

```
sp_configure "enable literal autoparam", [0 | 1]
```

Where 1 automatically converts literals to parameters, and 0 disables the feature. The default is 0.

You can set literal parameterization at the session level with the set command:

```
set literal_autoparam [off | on]
```

Examples

If you enable literal auto parameterization, the SQL text of the earlier `select count(*)` example queries is converted to:

```
select count(*) from titles where total_sales > @@@V0_INT
```

Where `@@@V0_INT` is an internally generated name for the parameter that represents the literal values 100 and 200.

All instances of literal values in the SQL text are replaced by internally generated parameters. For example:

```
select substring(name, 3, 4) from sysobjects where name in (@@@V0_VARCHAR,
@@@V1_VARCHAR)
```

is transformed to:

```
select substring (name, 3, 4) from sysobjects where name in (@@@V0_VARCHAR,
@@@V1_VARCHAR)
```

Any combination of values that replace the literals, `3`, `4`, `systypes` and `syscolumns` are transformed to the same SQL text with the same parameters and share the same query plan when you enable the statement cache.

Literal auto parameterization:

- Reduces compilation time on the second—and subsequent—executions of the query, regardless of the literal values in the query.
- Reduces storage space SQL text, including memory usage in the statement cache and the number of rows in sysqueryplans for abstract plans and query metrics.
- Reduces the amount of procedure cache used to store query plans.
- Occurs automatically within Adaptive Server, when enabled: you need not change the applications that submit the queries to Adaptive Server.

Usage issues for literal auto parameterization include:

- Adaptive Server parameterizes the literals only for select, delete, update, and insert. For insert statements, Adaptive Server parameterizes only insert ... select statements, not insert ... values statements.
- Adaptive Server does not parameterize queries similar to `select id + 1 from sysobjects group by id + 1` or `select id + 1 from sysobjects order by id + 1` because of the expressions ("`id + 1`") in the group by and order by clauses.
- Adaptive Server does not cache SQL statements with text longer than 16384 bytes in the statement cache. Transforming the literals in the SQL statement into variables can significantly expand the size of the SQL text (especially if there are a large number of literals). Enabling literal auto parameterization may result in Adaptive Server not caching some SQL statements that it would otherwise have cached.
- univarchar and unichar literals are not supported.
- If you are using a multibyte character set, enable literal autoparam is automatically disabled.
- If two SQL statements are the same except that their literal values have different datatypes, they are not transformed into matching SQL texts. For example, the following two SQL statements return the same results, but are parameterized differently because of the different datatypes:

  ```
  select name from sysobjects where id = 1
  select name from sysobjects where id = 1.0
  ```

  Their parameterized versions are:

```
select name from sysobjects where id = @@@V0_INT
select name from sysobjects where id = @@@V0_NUMERIC
```

System changes

Adaptive Server version 15.0.1 includes these new and changed functions, configuration parameters, commands, and stored procedures.

New functions

isdate

Description — Determines whether an input expression is a valid datetime value.

Syntax — isdate(*character_expression*)

Parameter — *character_expression* – is a character-type variable, constant expression, or column name.

Example 1 — Determines if the string 12/21/2005 is a valid datetime value:

```
select isdate('12/21/2005')
```

Example 2 — Determines if stor_id and date in the sales table are valid datetime values:

```
select isdate(stor_id), isdate(date) from sales
---- ----
0    1
```

store_id is not a valid datetime value, but date is.

Usage — Returns 1 if the expression is a valid datetime value; returns 0 if it is not. Returns 0 for NULL input.

isnumeric

Description — Determines if an expression is a valid numeric datatype.

Syntax — isnumeric (*character_expression*)

Parameter — *character_expression* – is a character-type variable, constant expression, or a column name.

Examples — Determines if the values in the postalcode column of the authors table contains valid numeric datatypes:

```
select isnumeric(postalcode) from authors
```

Determines if the value $100.12345 is a valid numeric datatype:

```
select isnumeric("$100.12345")
```

Usage

- Returns 1 if the input expression is a valid integer, floating point number, money or decimal type; returns 0 if it does not or if the input is a NULL value. A return value of 1 guarantees that you can convert the expression to one of these numeric types.
- You can include currency symbols as part of the input.

partition_object_id

Description — Displays the object ID for a specified partition ID and database ID.

Syntax — partition_object_id(*partition_id* [, *database_id*])

Parameter
- *partition_id* – ID of the partition whose object ID is to be retrieved.
- *database_id* – database ID of the partition.

Example 1 — Displays the object ID for the partition whose partition ID is 2:

```
select partition_object_id(2)
```

Example 2 — Displays the object ID for the partition whose partition ID is 14 and whose database ID is 7:

```
select partition_object_id(14,7)
```

Example 3 — Returns a NULL value for the database ID because a NULL value is passed to the function:

```
select partition_object_id( 1424005073, NULL)
 -----------
NULL
(1 row affected)
```

Usage
- partition_object_id uses the current database ID if you do not include a database ID.
- partition_object_id returns NULL if you use a NULL value for the *partition_id*.
- partition_object_id returns a NULL value if you include a NULL value for database ID.
- partition_object_id returns NULL if you provide an invalid or non-existent *partition_id* or *database_id*.

New configuration parameters

startup delay

Summary information	
Default value	1
Range of values	0 (disabled), 1 (enabled)
Status	Dynamic

Summary information	
Display level	Comprehensive
Required role	System Administrator
Configuration group	Query Tuning

startup delay controls when RepAgent is started during the server start. By default, RepAgent starts at the same time as Adaptive Server. Adaptive Server writes a message to the error log stating the wait time.

For example, this configures RepAgent to wait 50 seconds before starting:

```
sp_config_rep_agent pubs2,'startup delay','50'
```

enable literal autoparam

Summary information	
Default value	0
Range of values	1 (enabled), 0 (disabled)
Status	Dynamic
Display level	Intermediate
Required role	System Administrator
Configuration group	Query Tuning

enable literal autoparam enables and disables literal server-wide parameterization.

cis idle connection timeout

Summary information	
Default value	0
Range of values	0 – 2147483647
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Component Integration Services

cis idle connection timeout configures Adaptive Server to check for CIS connections to any remote server that have been unused longer than the specified number of seconds. Adaptive Server deletes the unused connections and reallocates their resources.

Although the number you specify is in seconds, the housekeeper task wakes up at most once a minute, so idle connections may be idle for much longer than the configured value. Adaptive Server does not drop idle connections if a transaction is active on the connection, and reestablishes the connection automatically if the user executes any command that accesses the connection.

enable logins during recovery

Summary information	
Default value	1
Range of values	0 (enabled), 1 (disabled)
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Security Related

enable logins during recovery determines whether non-System Administrator logins are allowed during database recovery. A value of 1 indicates that logins are allowed during recovery, and a value of 0 indicates that logins are not allowed during recovery, that is, only the System Administrator can log in to Adaptive Server).

sproc optimize timeout limit

Summary information	
Default value	40
Range of values	0 – 4000
Status	Dynamic
Display level	Comprehensive
Required role	System Administrator
Configuration group	Query Tuning

sproc optimize timeout limit specifies the amount of time Adaptive Server can spend optimizing a stored procedure as a fraction of the estimated execution time.

Changed configuration parameters

optimization timeout limit

The range of values available for the optimization timeout limit has changed. In earlier versions, the range of values was:

Summary information	
Range of values	1 – 1000

For Adaptive Server 15.0.1, the range of values is:

Summary information	
Range of values	0 – 1000

A value of 0 indicates there is no optimization timeout.

max parallel degree

If max parallel degree is set to 1 (enabled), Adaptive Server forces serial query execution and the optimizer may select plans with a higher parallel degree than if it is disabled.

number of worker processes

If you have not configured number of worker processes for a sufficient number of threads from the worker thread pool, Adaptive Server adjusts query plans at runtime to use fewer worker threads. If Adaptive Server cannot adjust the queries at runtime, the queries recompile serially. However, alter table and execute immediate commands are aborted if they do not have sufficient worker threads.

Changed commands

set command change

set literal_autoparam on|off

Enables and disables literal parameterization at the session level.

set opttimeoutlimit

The range of values for opttimeoutlimit has been changed to 0 – 4000, with 0 indicating no optimization limit.

set index_union on | off

When enabled, set index_union limits the scan of a table with an or clause.

Index unions (also known as an or strategy) are used for queries that contain or clauses. For example:

```
select * from titleauthor where au_id = "409-56-7008" or title_id = "PC8888"
```

If you have enabled index_union, this example uses an index on au_id to find the row IDs (RIDs) of all titleauthor tuples with `au_id = "409-56-7008"`, and uses an index on title_id to find the RIDs of all titleauthor tuples with `title_id = "PC8888"`. Adaptive Server then performs a union on all RIDs to eliminate duplicates. The resulting RIDs are joined with a RidJoin to access the data tuples.

If index_union is disabled, Adaptive Server does not use an index union strategy in a query to limit the table scan. Instead, it uses other access paths on the table (in the example above, it would use a table scan for table titleauthor), and applies the or clause as a filter in the scan operator.

alter table

Use the alter table command to drop one or more list or range partitions. You cannot use alter table to drop a hash or round-robin partition.

The syntax is:

alter table *table_name* drop partition
partition_name [, *partition_name*]...

For each partition you drop, Adaptive Server:

- Deletes all data on the partition.
- Deletes the partition definition from the system catalog.
- Drops all corresponding local index partitions that refer to this data partition.
- Regenerates the partition condition object of the base table and each local index.
- Deletes all statistics information on this partition.

- Rebuilds all global indexes.

Note If you attempt to drop a partition from a table that is referenced by another table, and the partition to be dropped and the referencing table are not empty, the command fails because of possible violations with the foreign-key constraint, and Adaptive Server displays error message 13971.

create index

When you create a unique local index on range-, list-, and hash-partitioned tables, the index key list is a superset of the partition-key list.

create existing table

create existing table includes syntax to determine whether an RPC uses the current or a separate connection:

```
create existing table (<column_list>) EXTERNAL [non_transactional
|transactional] PROCEDURE at 'location'
```

- `non_transactional` – a separate connection is used to execute the RPC.
- `transactional` – the existing connection is used to execute the RPC.

The default behavior is transactional.

update table statistics

Adaptive Server 15.0.1 adds the ability to run update statistics on a global index.

update table statistics *table_name*
[partition *data_partition_name*]
[*index_name* [partition *index_partition_name*]]

Where *index_name* is the name of index associated with the partition, and *index_partition_name* is the name of the index partition.

Because running update table statistics incurs the I/O cost of running update statistics, use update statistics to generate both column and table statistics.

You can create, and then drop, a global index to generate global statistics.

When you run update statistics on a single partition, you create global statistics by merging partition statistics. However, these merged global statistics are less accurate than the global statistics created as a side-effect of global index creation. Avoid generating column statistics that overwrite more accurate, earlier versions of column statistics.

Usage

- When *index_name* is specified, update table statistics updates statistics for all the index partitions of the index.
- When *index_partition* is specified, update table statistics updates statistics for the specific index partition.

New system procedures

sp_start_rep_agent

sp_start_rep_agent includes the recovery_foreground option to print progress when RepAgent starts in recovery mode.

Description

Starts a RepAgent thread for the specified database.

Syntax

sp_start_rep_agent dbname [, {'for_recovery' | 'recovery_foreground}'[, 'connect_dataserver', 'connect_database'[, 'repserver_name', 'repserver_username', 'repserver_password']]]]]

New parameter

recovery_foreground – configures RepAgent to print the recovery progress to the client.

Example

This configures RepAgent to print the recovery of database db2 to the client:

```
sp_start_rep_agent db2, recovery_foreground, ds, db1
RepAgent(5). Starting recovery, processing log records
between (1018, 0) and (2355, 2).
RepAgent(5). Processed 1000 log records.
RepAgent(5). Processed 2000 log records.
RepAgent(5). Processed 3000 log records.
RepAgent(5). Processed 4000 log records.
RepAgent(5). Processed 5000 log records.
RepAgent(5). Processed 6000 log records.
RepAgent(5). Processed 7000 log records.
RepAgent(5). Processed 8000 log records.
RepAgent(5). Processed 9000 log records.
RepAgent(5). Processed 10000 log records.
RepAgent(5). Processed 11000 log records.
RepAgent(5). Processed 12000 log records.
RepAgent(5). Processed 13000 log records.
```

```
RepAgent(5). Processed 14000 log records.
RepAgent(5). Processed 15000 log records.
RepAgent(5). Processed 16000 log records.
RepAgent(5). Processed 17000 log records.
RepAgent(5). Processed 18000 log records.
RepAgent(5). Processed 19000 log records.
RepAgent(5). Processed 20000 log records.
RepAgent(5). Processed 20084 log records, recovery
complete.
Replication Agent thread is started for database 'db2'.
(return status = 0)
```

Changed system procedures

sp_setreptable

sp_setreptable now includes a use_index parameter.

Syntax

```
sp_setreptable <table_name> [, true | false]
[, owner_on | owner_off | null]
[, use_index]
```

New parameter

use_index – Adaptive Server creates a global nonclustered index on the text or image pointer for all existing tables with text or image data that are not explicitly marked for replication.

Example

Creates table t1, marks the table for replication, and creates two indexes on the text and image pointer for the columns t and i.

```
create table t1 (c int, t text, i image)
go
sp_setreptable t1, true, null, use_index
go
```

Marks the table for replication without using indexes:

```
sp_setreptable t1, true
```

sp_setrepcol

sp_setrepcol now includes a use_index parameter.

Syntax

```
sp_setrepcol <tab_name> [, column_name] [, do_not_replicate |
replicate_if_change |
always_replicate]
[, use_index]
```

New parameter — use_index – creates an index for replication on the text or image columns.

Examples — Creates an index for replication on the t column:

```
create table t1 (c int, t text, i image)
go
sp_setreptable t1, true
go
sp_setrepcol t1, t, replicate_if_changed, use_index
go
```

Usage

- The order of the precedence on the index status is column, then table, then database.
- If the database is set to use indexes, but the object is marked to not use them, the object status overrides the database status. For example, in a database marked for replication using indexes, the table's status overrides the index status. Similarly, if you set a table to use the indexes, some columns may not use the indexes.

sp_reptostandby

sp_reptostandby now includes a use_index parameter.

Syntax

```
sp_reptostandby <db_name> [,'ALL' | 'NONE' | 'L1']
[, 'use_index']
```

New parameter — use_index – Adaptive Server creates a global nonclustered index on the text or image pointer for all existing tables with text or image data that are not explicitly marked for replication.

Example — Sets the replication status for pubs2 to ALL and creates a global index on the text and image pointers:

```
sp_reptostandby pubs2,'ALL', 'use_index'
```

Changed monitoring tables

Table 21-1 describes updates to monitoring tables.

Table 21-1: Updates to monitoring tables

Column name	Monitoring table	Description
RowsAffected	monSysStatement	Indicates the number of rows affected by the current statement. Queries using an inefficient query plan probably show a high number of logical I/Os per returned row.
ErrorStatus	monSysStatement	The error return status of the statement.

Column name	Monitoring table	Description
RowsAffected	monProcessStatement	Indicates the number of rows affected by the current statement. Queries using an inefficient query plan probably show a high number of logical I/Os per returned row.
BlockedBy	monLocks	Identifies the lock ID blocking this lock request. Zero indicates no block.
BlockedState	monLocks	Identifies lock status.

New monitoring tables

For complete information about monitoring tables, see the *Performance and Tuning Guide: Monitoring and Analyzing*.

monProcedureCacheMemoryUsage

Description

Has one row for each procedure cache allocator. An allocator is identified by an allocator ID, which is internal to Adaptive Server.

Columns

Table 21-2: monProcedureCacheMemoryUsage

Name	Datatype	Attributes	Description
Allocator ID	int		An allocator ID
ModuleID	int		A module ID (internal to Adaptive Server)
Active	int		Number of memory pages (2KB) currently allocated to this allocator
HWM	int		The maximum number of memory pages allocated since the server was started
ChunkHWM	int		The largest number of contiguous memory pages allocated since the server was started
AllocatorName	varchar(30)		Name of the allocator
NumReuseCaused	int	null	Number of times this allocator has caused replacement

monProcedureCacheModuleUsage

Description

Has one row for each module that allocates memory from procedure cache. A module, which is identified with a module ID, is a functional area classification internal to Adaptive Server procedure cache management.

Columns

Table 21-3: monProcedureCacheModuleUsage

Name	Datatype	Attributes	Description
ModuleID	int		A module ID
Active	int		Number of memory pages (2KB) currently allocated to this module
HWM	int		The maximum number of memory pages allocated since the server was booted
ModuleName	int		Name of the module
NumPagesReuse	varchar(30)	null	Number of pages allocated to this module

Feature and platform compatibility

Table 21-4 shows feature availability for supported operating systems in Adaptive Server version 15.0.1. “Y” indicates the feature is supported for that platform.

Table 21-4: Adaptive Server features for supported operating systems

Adaptive Server options	Solaris 32-bit	Solaris 64-bit	HP-UX PA Risc 64-bit	IBM AIX 64-bit	Linux x86 32-bit	Windows x86 32-bit	Linux on Power 64-bit	Linux Opteron 64-bit	Sol Opteron 64-bit	HP-UX Itanium 64-bit	Windows Opteron X64
Security and directory services	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Cybersafe Kerberos	Y	Y				Y					
Pluggable Authentication Module	Y	Y		Y	Y			Y	Y	Y	
Fine grained access control	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LDAP server directory	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y
LDAP user authentication	Y	Y	Y	Y	Y	Y		Y	Y	Y	
Platform Native Kerberos	Y	Y									
Secure Sockets Layer	Y	Y	Y	Y	Y	Y		Y	Y	Y	
MIT Kerberos	Y	Y	Y	Y	Y		Y	Y	Y	Y	

Adaptive Server options	Solaris 32-bit	Solaris 64-bit	HP-UX PA Risc 64-bit	IBM AIX 64-bit	Linux x86 32-bit	Windows x86 32-bit	Linux on Power 64-bit	Linux Opteron 64-bit	Sol Opteron 64-bit	HP-UX Itanium 64-bit	Windows Opteron X64
Encrypted columns	Y	Y	Y	Y	Y	Y		Y	Y	Y	
High availability	Y	Y	Y	Y	Y	Y				Y	
Partitions	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Enhanced Full-Text Search (EFTS)	Y	Y	Y	Y	Y	Y					
Features included with Adaptive Server											
Cross-platform dump and load	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Job Scheduler	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
ASE Replicator	Y	Y	Y	Y	Y	Y					
IPv6	Y	Y	Y		Y	Y					
Java option	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Native XML	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Java-based XML	Y	Y	Y	Y	Y	Y				Y	
Web Services	Y	Y	Y	Y	Y	Y		Y	Y	Y	
Distributed Transaction	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Content Management	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

PART 3

New Features for Adaptive Server Version 12.5.4

Part 3 includes these feature descriptions:

CHAPTER 22

Feature and platform compatibilty for Adaptive Server 12.5.4

This chapter includes a feature and platform compatibility matrix, as well as summary information about the new features introduced in Adaptive Server versions 12.5.4, 12.5.3a, and 12.5.3.

Feature and platform compatibility information

Table 22-1 shows feature availability for supported operating systems in Adaptive Server version 12.5.4.

Table 22-1: Adaptive Server features for supported operating systems

Adaptive Server options	Solaris 32-bit	Solaris 64-bit	Solaris x86	Sol Opteron 64-bit	HP-UX PA Risc 64-bit	HP-UX PA Risc 32-bit	HP Tru64	HP-UX Itanium 64-bit	IBM AIX 32-bit	IBM AIX 64-bit	Windows x86	Macintosh OS X	SGI 32-bit	SGI 64-bit	Linux on pSeries	Linux Opteron 64-bit	Linux Itanium 64-bit	Linux x86 32-bit
Encrypted columns	*f*	*f*	–	–	*f*	*f*	–	–	–	*f*	*f*	–	–	–	–	–	–	*f*
High availability	Y	Y	–	–	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	–	–	–	Y
Distributed Transaction Management	Y	Y	–	Y	Y	Y	–	–	Y	Y	Y	–	Y	Y	–	Y	–	Y
XML management	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	–	Y	Y
Java option	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y

Legend: Y: supported in Adaptive Server 12.5.x; *f*:Introduced in 12.5.3a; ž:Introduced in12.5.4; §: Introduced in 12.5.3 ESD5; Þ: port merge; **–: Not supported**

Adaptive Server options	Solaris 32-bit	Solaris 64-bit	Solaris x86	Sol Opteron 64-bit	HP-UX PA Risc 64-bit	HP-UX PA Risc 32-bit	HP Tru64	HP-UX Itanium 64-bit	IBM AIX 32-bit	IBM AIX 64-bit	Windows x86	Macintosh OS X	SGI 32-bit	SGI 64-bit	Linux on pSeries	Linux Opteron 64-bit	Linux Itanium 64-bit	Linux x86 32-bit
Native XML	Y	Y	–	Y	Y	Y	–	§	–	Y	Y	–	–	–	–	§	–	Y
Java-based XML	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	–	Y	Y
Web Services	Y	Y	–	–	Y	Y	–	Y	Y	Y	Y	–	Y	Y	–	–	Y	Y
Security and directory services	Y	Y	–	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y
LDAP server directory	Y	Y	–	Y	Y	Y	Y	Y	Y	*ƒ*	Y	–	Y	–	–	–	ž	Y
LDAP user authentication	Y	Y	–	–	Y	Y	Y	–	Y	*ƒ*	Y	–	–	–	–	–	–	Y
Secure Sockets Layer	Y	Y	–	Y	Y	Y	Y	ž	Y	Y	Y	–	–	–	–	–	–	Y
Cybersafe Kerberos	Y	Y	–	–	–	–	–	–	Y	–	Y	–	–	–	–	–	–	–
MIT Kerberos	Y	Y	–	Y	*ƒ*	–	–	–	*ƒ*	*ƒ*	–	–	–	–	–	Y	–	Y
Platform Native Kerberos	Y	Y	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Fine grained access control	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y
Pluggable Authentication Module	Y	Y	–	Y	–	–	–	ž	–	*ƒ*	–	Y	–	–	–	Y	Y	Y
Content management (eXternal file support)	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y
Monitor Client GUI	–	–	–	–	–	–	–	–	–	–	Y	–	–	–	–	–	–	–
Enhanced Full-Text Search (EFTS)	Y	Y	–	–	Y	Y	Y	–	Y	Y	Y	–	–	–	–	–	–	Y
Real-time messaging	Y	Y	–	–	–	Y	–	–	–	Y	Y	–	–	–	–	–	–	Y
JMS support	Y	Y	–	–	–	Y	–	–	Y	Y	Y	–	–	–	–	–	–	Y
WebSphere MQ support	*ƒ*	*ƒ*	–	–	–	*ƒ*	–	–	–	*ƒ*	Y	–	–	–	–	–	–	*ƒ*
Disaster recovery	Y	–	–	–	Y	–	Y	Y	Y	–	Y	–	Y	–	–	–	–	Y
Features included in base Adaptive Server																		
Password complexity	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž
Archive database access	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž	ž
IPv6	Y	Y	–	–	*ƒ*	*ƒ*	–	–	–	–	*ƒ*	–	–	–	–	–	–	–
XA libraries	–	–	–	Y	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Cross-platform dump and load	Y	Y	–	Y	Y	Y	–	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Job Scheduler	Y	Y	–	Y	Y	Y	–	–	Y	Y	Y	Y	Y	Y	–	–	–	Y
ASE Replicator	Y	Y	–	–	Y	Y	–	–	Y	Y	Y	Y	–	–	–	–	–	Y

Legend: Y: supported in Adaptive Server 12.5.x; *ƒ*:Introduced in 12.5.3a; ž:Introduced in12.5.4; §: Introduced in 12.5.3 ESD5; Þ: port merge; **–: Not supported**

New feature overview

Table 22-2 includes information about the major features included in Adaptive Server versions 12.5.3, 12.5.3a, and 12.5.4. For detailed descriptions of these features.

Table 22-2: New features for Adaptive Server Enterprise version comparison

Adaptive Server feature	Description
Adaptive Server version 12.5.4	
Kerberos enhancements	Adaptive Server includes the following Kerberos enhancements: • New option to specify a Kerberos principal name different from the Adaptive Server name. • Expanded platform support for Kerberos. • You can now use the sp_modifylogin option authenticate with to require Kerberos authentication for an individual login. • sybmapname is a customizable utility to convert external user principal names to the name space of Adaptive Server logins.
LDAP user authentication enhancements	Adaptive Server includes the following LDAP user authentication enhancements: • Automatic failover from authentication against a primary LDAP server to a secondary LDAP server. • Adaptive Server recovers from errors encountered when communicating with the LDAP server. • Enhancements to the communication of password-expiration-related messages obtained from the LDAP server to Adaptive Server clients.
Password complexity enhancements	You can now specify a wide range of password complexity attributes. You can also write your own stored procedures to create site-specific password complexity checks.
Archive database access support	Archive database access provides a database administrator with the capability to validate or selectively recover data from a database dump by making the dump appear as if it were a traditional read-only database.
Shared directory changes	Adaptive Server version 12.5.4 includes a number of changes to the shared directory structure.
Sybase driver support	Adaptive Server now includes the following Adaptive Server data providers: • ASE ADO.NET Data Provider 1.1 • ASE OLE DB Provider by Sybase 12.5.1 • ASE ODBC Driver by Sybase 12.5.1

Adaptive Server feature	Description
Dynamically loading TIBCO libraries	TIBCO JMS libraries are now dynamically loaded rather than statically linked to the Adaptive Server executable.
JRE support	Adaptive Server version 12.5.4 includes JRE 1.4.
Adaptive Server plug-in changes	Adaptive Server plug-in supports archive database access and password complexity.
Updating system catalogs	The server-wide command allow updates to system catalogs takes precedence over the stored procedure settings for allow updates.
Monitoring tables changes	Monitoring tables monSysStatement andmonProcessStatement have been changed.
syscomments changes	When the source text of a stored procedure or trigger is stored in the system table syscomments, a query using select * is stored in syscomments expanding the column-list referenced in the select *.
Shared memory support in Windows terminal server environments	To accommodate Windows terminal server shared memory requirements, Adaptive Server version 12.5.4 introduces the new environment variable SYBASE_TS_MODE.
Global login trigger	Adaptive Server version 12.5.4 provides the ability to set a new global login trigger that is executed at each user login.
Exporting set options from a login trigger	Adaptive Server version 12.5.4 enables options set inside login triggers to remain valid for the entire user session.
Adaptive Server version 12.5.3a	
Encrypted columns support	Data encryption in Adaptive Server allows you to encrypt data at the column level.The authentication and access control mechanisms ensure that only properly identified and authorized users can access data. The encrypted columns feature in Adaptive Server is easier to use than encryption in the middle tier or in the client application. No application changes are required when enabling encrypted columns support.
Adaptive Server version 12.5.3	
Secure Sockets Layer enhancements	New SSL CipherSuites supporting Advanced Encryption Standard (defined in FIPS-197) for encrypting network communications. Advanced Encryption Standards allow you to choose whether to use 128- or 256-bit encryption keys.
Support for cross-platform dump and load for databases	Adaptive Server supports databases dumps and loads across platforms with different endian architecture. You can perform dump database and load database from either a big-endian platform to a little-endian platform, or from a little-endian platform to a big-endian platform.
Importing statistics for proxy tables	If the relevant tables and index statistics are available when you perform update statistics on a remote server proxy table, the table catalogs are imported to the local systabstats and sysstatistics.

Adaptive Server feature	Description
top n functionality support	Use the top n clause to limit the number of rows in the result set to the number of rows specified by the integer. Adaptive Server Enterprise supports the top n clause in outer query select statements, but not in the select list of a subquery.
Historical Server enhancements	Adaptive Server allows you to send monitoring data from Historical Server to a database on a specific Adaptive Server.
Real-time data services enhancements	Use sp_configure to set the number for native threads and the wait time for messaging.
Resource governor	Adaptive Server resource limits to help System Administrators prevent queries and transactions from monopolizing server resources. In Adaptive Server version 12.5.3, when System Administrators modify a resource limit, all users logged in the session see the change.
Page allocation for partitioned DOL tables	Adaptive Server version 12.5.3 avoids using extra space by filling up existing allocated extents in the target allocation page even when these extents are assigned to other partitions.
User connections enhancements	User connection enhancements include an updated error message and a correction on reserved sockets.
dtdValidation	The dtdValidate option include three additional parameters: no, yes, and strict.
sybmigrate enhancements	The sybmigrate tool allows you to migrate across versions of Adaptive Server Enterprise and supports source servers from 12.0 and later.
New Enhanced Full-Text Search Specialty Data Store language support	EFTS now includes language support for: • Traditional Chinese on the Windows and Solaris platforms • Arabic, Hebrew, Thai, and Russian on the Linux platform
Monitor counters and sp_sysmon	Adaptive Server version 12.5.3 introduces a number of enhancements to improve some of the most commonly used monitoring tools.
Adaptive Server version 12.5.2 (for information, see the 12.5.2 New Feature Guide)	
Pluggable Authentication Module support	Pluggable Authentication Modules (PAM) support allows multiple authentication service modules to be stacked and made available without modifying the applications that require the authentication.
Kerberos native library support	Adaptive Server provides the following Kerberos native library support: • Cybersafe Library support • MIT Kerberos library support • Active directory interoperability for user authentication • Kerberos support in jConnect

Adaptive Server feature	Description
Support for password-protected backups	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.
Access control enhancements	Adaptive Server introduces the following access control enhancements: • Restricted permissions on system catalogs • Improved granularity for set proxy • Grant and revoke for administration commands.
FIPS-140-certified algorithms support	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.
Statement cache enhancements	The statement cache is used for saving SQL cached-statements.
XML services support	The changes in XML services enhance four main areas: • Extended XML query language for the xmlextract built-in function and xmltest predicate. • Enhanced datatype support in the for xml clause of select statements. • Enhanced datatype support for xmlextract, xmlparse, and xmlrepresentation. • Enhanced Java-XML sample code.
Real-time messaging services	Adaptive Server 12.5.2 includes 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.
Web services support	A Web service is a self-contained, modular application that can be accessed through a network connection. Using a Web service, you trade performance for increased interoperability.
IPv6 support	Adaptive Server supports Internet Protocol version 6 (IPv6.)
Enhanced Full-Text Search (EFTS) enhancements	EFTS introduces the following enhancements: • Installation and directory changes • Permission for shutdown • index_any clauses support up to 16000 bytes • New pseudo-column total_docs • Primary keys
Compressed database dump support	The Adaptive Server version 12.5.2 version of dump includes a compression parameter that allows you to compress your database dumps.

Adaptive Server feature	Description
Password-protecting database dumps and loads	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.
Failover on Veritas 2.1 support	You can now configuring Adaptive Server on Linux platforms for Failover on Veritas 2.1.
Large memory support	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.

CHAPTER 23

Dynamically Loading TIBCO Libraries

This chapter describes Adaptive Server support for dynamically loaded TIBCO JMS libraries.

Overview

Adaptive Server version 12.5.4 dynamically loads the messaging libraries it needs to interact the with TIBCO EMS and IBM MQ message buses.

The Adaptive Server messaging libraries contain the messaging logic and act as a wrappers on top of the messaging libraries provided by the vendors like TIBCO and IBM. These Adaptive Server messaging libraries are available with the purchase of Real Time Data Services (RTDS) 3.5 and later. After you install Adaptive Server, you must install RTDS 3.5 to install the Adaptive Server messaging DLLs. Install these libraries in the *$SYBASE/ASE-12_5/lib* directory.

In addition to Adaptive Server Messaging libraries, you must have the messaging libraries and DLLs from vendors (TIBCO and IBM.)

Once you have installed Adaptive Server Messaging libraries and the vendor-specific message bus libraries, modify the LD_LIBRARY_PATH or the equivalent with the location of these libraries.

Adaptive Server Messaging libraries are located in *$SYBASE/ASE-12_5/lib*, by default.

You must also add the location of the vendor provided DLLs in the LD_LIBRARY_PATH or the equivalent.

Adding TIBCO JMS DLL location information

Assuming TIBCO_HOME is the location where TIBCO EMS version 4.2 or 4.3 have been installed, the platform-specific paths are as follows:

- On Solaris 32-bit platforms, add *$TIBCO_HOME/clients/c/lib* to LD_LIBRARY_PATH
- On Solaris 64-bit platforms, add to following to LD_LIBRARY_PATH:
  - *$TIBCO_HOME/clients/c/lib/64*
  - *$TIBCO_HOME/clients/c/lib*
- On Linux 32-bit platforms, add *$TIBCO_HOME/clients/c/lib* to LD_LIBRARY_PATH.
- On Windows 32-bit platforms, add *%TIBCO_HOME%\clients\c\bin* to PATH.
- On HP-UX 64-bit platforms, add the following to LD_LIBRARY_PATH:
  - *$TIBCO_HOME/clients/c/lib/64*
  - *$TIBCO_HOME/clients/c/lib*
- On IBM AIX 64-bit platforms, add the following to LIBPATH:
  - *$TIBCO_HOME/clients/c/lib/64*
  - *$TIBCO_HOME/clients/c/lib*

Adding IBM MQ DLLs to LD_LIBRARY_PATH

Assuming MQM_HOME is where the MQ client library is installed, the platform-specific paths are as follows:

- On Solaris 32-bit platforms, add *$MQM_HOME/lib* to LD_LIBRARY_PATH
- On Solaris 64-bit platforms, add *$MQM_HOME/lib64* to LD_LIBRARY_PATH
- On Linux 32-bit platforms, add *$MQM_HOME/lib* to LD_LIBRARY_PATH.
- On Windows 32-bit platforms, add *%MQM_HOME%\bin* to PATH.

- On HP-UX 64-bit platforms, add *$MQM_HOME/lib64* to LD_LIBRARY_PATH.
- On IBM AIX 64-bit platforms, add *$MQM_HOME/lib64* to LIBPATH.

Note MQM_HOME is */opt/mqm* for Solaris, Linux, and HP-UX, and */usr/mqm* on IBM AIX. On Windows it is the directory where Websphere MQ is installed.

Additionally, sp_configure 'enable real time messaging' has been modified to accept the type of messaging that you want to enable:

- `sp_configure 'enable real time messaging', 1` enables both TIBCO JMS and IBM MQ messaging. This command succeeds if Adaptive Server can locate the DLL libraries for TIBCO JMS and IBM MQ.
- `sp_configure 'enable real time messaging', 1, 'tibco_jms'` enables TIBCO JMS messaging only. This command succeeds if Adaptive Server can locate the DLL libraries for TIBCO JMS.
- `sp_configure 'enable real time messaging', 1, 'ibm_mq'` enables IBM MQ messaging only. This command succeeds if Adaptive Server can locate the DLL libraries for IBM MQ

CHAPTER 24

Shared Memory in Windows Terminal Server Environments

Shared memory in the Windows Terminal Server

To accommodate Windows Terminal Server shared memory requirements, Adaptive Server version 12.5.4 introduces the environment variable SYBASE_TS_MODE.

Windows Terminal Server is a feature of Microsoft Windows that allows multiple remote users to log into the Windows system simultaneously. Each user connects from a local workstation to a central Windows server, and is then given a virtual Windows environment which appears back on the local workstation. The Windows server keeps the memory spaces and other resource allocations of these terminal server sessions entirely separate, so that users are largely unaware of the existence of other terminal server sessions. In addition, there can be a user logged in to the machine in the normal way–this is referred to as the “console session”.

The Adaptive Server data server can be started in a terminal server session, or it may start as a Windows service, in which case it is regarded as part of the console session. Adaptive Server shared memory regions are normally not accessible from other terminal server sessions, and thus certain tools such as performance monitors, and the Sybase internal diagnostic tool (sybmon), will not work unless started from the same session that owns the Adaptive Server, as these tools must connect to the shared memory region of that server. If Adaptive Server is running as a service, these tools work only if started from the console session.

Adaptive Server 12.5.4 looks for an environment variable named SYBASE_TS_MODE, and if it is set to GLOBAL then Adaptive Server creates its shared memory so that it is available across all terminal server sessions. Similarly, changes made to the performance monitor and sybmon also look for this environment variable and connect to the shared memory globally if it is set to GLOBAL. In this way, you can configure a remote terminal session to monitor Adaptive Server running on the console session or on another terminal server session. Create SYBASE_TS_MODE as a system environment variable then restart the Windows machine to bring it into effect.

SYBASE_TS_MODE is not supported by Windows NT 4.0 Workstation and Server, and Windows 2000 Professional.

If SYBASE_TS_MODE is not set, shared memory is created only under the local terminal server session.

Warning! Do not set SYBASE_TS_MODE on a machine that does not support terminal services. This causes shared memory creation to fail, and Adaptive Server will not start.

CHAPTER 25

Security Enhancements

This chapter discusses security enhancements in Adaptive Server Enterprise.

Kerberos changes

Kerberos has been enhanced in the following ways:

- You can now specify a server principal name for Kerberos authentication
- The sp_modifylogin and sp_addlogin authenticate with options support Kerberos authentication
- Support for sybmapname
- Support for the MIT Kerberos ClientLibrary
- Expanded platform support for Kerberos

Specifying the Adaptive Server principal name

The principal name is the name the server uses to authenticate with the Kerberos Key Distribution Center (KDC). When you have multiple instances of Adaptive Server running, you must have different principal names for each Adaptive Server.

In Adaptive Server version 12.5.4, you can use a new dataserver option and a new environment variable to specify a principal name different from the Adaptive Server name. Adaptive Server name is specified by environment variables DSLISTEN and DSQUERY, or the dataserver command-line option `"-s servername"`.

You can set the principal name either the setenv command or the -k dataserver option.

By default, the principal name is the name of Adaptive Server. To specify a different name, set SYBASE_PRINCIPAL before starting Adaptive Server to use Kerberos:

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

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

You can use the following command-line option to specify an Adaptive Server principal name, when starting Adaptive Server:

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

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

Example

In this example, the Adaptive Server name is “ase1254” and realm name is “MYREALM.COM,” the Adaptive Server name is specified on the command line with -s parameter to the dataserver. The current realm is specified in *libtcl.cfg* by a secbase attribute value:

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

The default Adaptive Server principal name is "ase1254@MYREALM.COM." If the principal name defined in the Adaptive Server *keytab* file is "aseprincipal@MYREALM.COM," you can override the default Adaptive Server principal name by setting a server principal name using options 1 or 2 below:

- Option 1: -k is specified:

  ```
  %
  $SYBASE/$SYBASE_ASE/bin/dataserver -dmaster.dat
  -s ase1254 -k aseprincipal@MYREALM.COM
  ```

  The Adaptive Server principal name used to authenticate with Kerberos is "aseprincipal@MYREALM.COM."

- Option 2: -k is not specified but SYBASE_PRINCIPAL is set:

  ```
  setenv SYBASE_PRINCIPAL aseprincipal@MYREALM.COM
  $SYBASE/$SYBASE_ASE/bin/dataserver –dmaster.dat
  -s ase1254
  ```

  The Adaptive Server principal name used to authenticate with Kerberos is "aseprincipal@MYREALM.COM," the value of *$SYBASE_PRINCIPAL*.

- Option 3: Neither -k nor SYBASE_PRINCIPAL is set

  ```
  % $SYBASE/$SYBASE_ASE/bin/dataserver –dmaster.dat
  -s ase1254
  ```

  The Adaptive Server principal name used to authenticate with Kerberos is "ase1254@MYREALM.COM."

For more information about Kerberos, see the Security section of the *System Administration Guide, Volume One*.

sp_modifylogin and *sp_addlogin authenticate with* option

In Adaptive Server version 12.5.4, the authenticate with option to sp_modifylogin or sp_addlogin requires that the login use *only* a specified authentication mechanism. The supported authentication mechanisms are:

- ASE
- LDAP
- PAM
- KERBEROS

- ANY

Using authenticate with with these supported authentication mechanisms allows you to override the server-wide configuration parameters unified login required, enable ldap user auth, and enable pam user auth.

If more than one external authentication mechanism is configured and a login-specific authenticate with option has not been set, the external authentication mechanism is based on the following order:

1 Kerberos

2 LDAP

3 PAM

For example, if both PAM and LDAP are configured, LDAP is chosen for external authentication, not PAM.

At the time of authentication, exactly one external authentication mechanism is attempted. Depending on the value of configuration parameters, you can try the "ASE" authentication mechanism when the external authentication fails.

When none of the external authentication mechanisms are configured, Adaptive Server uses ASE authentication.

Using *authenticate with* to override server-wide authentication options

Note You must have sso_role permissions to modify the authenticate with option for a login.

To configure external authentication mechanisms such as Kerberos, LDAP, or PAM, Sybase recommends that you determine the server-wide authentication setting that matches your company's security policy. This server-wide setting is appropriate for most client connections. Then, you can set individual logins to another authentication mechanism using the authenticate with option.

You can use authenticate with to specify an the external authentication mechanisms Kerberos, LDAP, and PAM. You can also issue sp_modifylogin or sp_addlogin authenticate with to set the authentication mechanism to "ASE" to use only the Adaptive Server internal authentication mechanism. To allow any authentication mechanism, use authenticate with ANY.

Note When authentication mechanism "ANY" (the default) is set for a login, the login uses the server-wide configuration settings to control authentication.

sp_modifylogin also checks for any conflicts with any login mapping specified by a previous sp_maplogin. See "Adding tighter controls on login mapping" on page 363 for more details.

Examples

Example 1 This example creates a local account to run a batch application. Consider an environment that uses Kerberos for a centralized user account repository and requires its general population of users to authenticate using Kerberos, Configure Kerberos by setting these parameters:

```
sp_configure  "use security services", 1
go

sp_configure "unified login required", 1
go
```

These configuration parameters now require all user logins, other than "sa," to authenticate using Kerberos to gain access to the Adaptive Server.

Now consider a nightly batch operation run by the Adaptive Server database administrator or operator, which may authenticate locally without requiring the account to exist in the Kerberos repository. Use authenticate with option to sp_modifylogin or sp_addlogin:

```
sp_addlogin nightlybatch, localpassword, ...
go
sp_modifylogin nightlybatch, 'authenticate with', 'ASE'
go
```

Example 2: Migrating users from Adaptive Server authentication to LDAP user authenication This example uses a phased approach for moving clients from local Adaptive Server authentication to LDAP user authentication.The LDAP directory server has been set up but has not yet been populated with all user accounts. A small population of users has agreed to a pilot program to test out LDAP for external authentication to Adaptive Server. To allow failover to ASE authentication when authentication with the LDAP directory server fails or the LDAP server is unavailable, enter:

```
sp_configure 'enable ldap user auth', 1
go
```

Users without LDAP accounts fail over to Adaptive Server authentication. The users in the pilot program are automatically added to the LDAP directory server and may begin authenticating using the LDAP directory server.

A user can use the *@@authmech* global variable to determine which authentication mechanism was used to authenticate with with:

```
select @@authmech
```

You can require users in the pilot program to use only LDAP authentication by entering:

```
sp_maplogin loginame, 'authenticate with', 'ldap'
go
```

If you decide to use LDAP user authentication for all users, change the configuration parameter to level 2. Any logins set to authenticate with LDAP during the pilot program can be reset to the default value "ANY". Because the configuration parameter is set to 2. "ANY" logins are still required to use LDAP:

```
sp_configure 'enable ldap user auth', 2
go
sp_maplogin loginame, 'authenticate with', 'any'
go
```

Note If a login has authenticate with set to a specific authentication mechanism such as LDAP, Kerberos, PAM, or ASE, that login can use only that mechanism for authentication. Sybase suggests that you use authentication with to set exceptions to the server-wide settings and to force a particular authentication mechanism to be used.

Using *sybmapname* to handle user principal names

The purpose of sybmapname is to convert external user principal names used in Kerberos environment to the namespace of Adaptive Server user logins. sybmapname is a customizable shared object that can map names given on its input buffer to a name suitable for Adaptive Server login on its output buffer.

You can use sybmapname shared object to perform the custom mapping between the user principal name and the Adaptive Server login name. This shared object is optionally loaded at server start-up, and the function syb__map_name contained in the shared object is called after a successful Kerberos authentication and just before the user principal is mapped to a login in the syslogins table. This function is useful when the user principal name and the login name to be mapped are not identical.

The customizable logic is the function:

```
syb__map_name(NAMEMAPTYPE *protocol, char *orig,
int origlen, char *mapped, int *mappedlen)
```

Where:

- `NAMEMAPTYPE *protocol` – refers to a structure reserved for usage of this function.
- `char *orig` – is an input buffer that is not null-terminated.
- `int origlen` – is the input buffer length. It should be less than or equal to 255 characters.
- `char *mapped` – is an output buffer that should not be null-terminated.
- `int *mappedlen` – is an output buffer length. It should be less than or equal to 30.

The function returns a value greater than 0 if the mapping succeeds, it returns a value of 0 if no mapping occurred, and it returns a value less than 0 when an error occurs in syb__map_name(). When an error occurs, a message displays in the Adaptive Server error log to report the mapping failure.

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

1 Configure Adaptive Server to use the Kerberos security mechanism. See the Adaptive Server *System Administration Guide* and Open Client/Server documentation, and the white paper titled “Configuring Kerberos for Sybase” on the Sybase Web site at http://www.sybase.com/detail?id=1029260.

   A sample *sybmapname.c* file is located in *$SYBASE/$SYBASE_ASE/sample/server/sybmapname.c*.

2 Modify *sybmapname.c* to implement your logic. Take precautions while coding as it may interfere with the proper running of Adaptive Server. See “Precautions when using sybmapname” on page 347.

3 Build the shared object or DLL using the generic platform specific makefile supplied. The makefile may need to be modified to suit your platform specific settings.

4 Place the resulting shared object generated in a location specified in your $LD_LIBRARY_PATH on UNIX machines, and PATH variable on Windows machines. The file should have read and execute permissions for the sybase user.

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

Verifying your login to Adaptive Server using Kerberos authentication

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

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

Example 1 If a client's principal name is `user@REALM`, and the corresponding entry in syslogins table is `user_REALM`, then sybmapname can be coded to accept the input string `user@realm` and to convert the input string to the output string `user_REALM`.

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

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

The following actions and output further illustrate the sybmapname function described in Example 2. The *sybmapname.c* file containing the customized definition for syb__map_name() should be compiled and built as a shared object (or DLL), and finally placed in the appropriate path location. Start Adaptive Server with the Kerberos security mechanism enabled.

To initialize the Ticket Granted Ticket (TGT):

```
$ /krb5/bin/kinit johnd@public
Password for johnd@public:
```

```
$
```

To list the TGT:

```
$ /krb5/bin/klist
   Cache Type: Kerberos V5 credentials cache
   Cache Name: /krb5/tmp/cc/krb5cc_9781
Default principal: johnd@public
```

Log in as "sa" and verify the user login for "johnd":

```
$ $SYBASE/$SYBASE_OCS/bin/isql -Usa -P
      -Ipwd`/interfaces
1>

1> sp_displaylogin johnd
2> go
No login with the specified name exists.
(return status = 1)

1> sp_displaylogin JOHND
2> go
Suid: 4
Loginame: JOHND
Fullname:
Default Database: master
Default Language:
Auto Login Script:
Configured Authorization:
Locked: NO
Password expiration interval: 0
Password expired: NO
Minimum password length: 6
Maximum failed logins: 0
Current failed login attempts:
Authenticate with: ANY
(return status = 0)
```

Successful Kerberos authentication, which maps lower case `johnd` to uppercase `JOHND` using the sybmapname utility and allows user `johnd` to log in to Adaptive Server:

```
$ $SYBASE/$SYBASE_OCS/bin/isql -V -I'pwd'/interfaces
1>
```

Precautions when using *sybmapname*

Be aware of the following issues when coding for sybmapname:

- Take care with the sample *sybmapname.c* program and any modifications to it. Avoid using code that may create a segmentation fault, that may call exit(), that may call system calls(), that may change UNIX signals, or that makes any blocking calls. Improper coding or calls may interfere with the Adaptive Server engine.

  Note Sybase bears no responsibility for coding errors in sybmapname.

- Code defensively, check all pointers before dereferencing them, and avoid system calls. The functions you write must be quick name-filtering functions.
- Do not use goto statements since, depending on the platform, they may cause unexpected side effects.
- If you use multiple realms, take care to map the user principal names to a suitable login name to reflect the realm information. For example, if you have two users whose user principal names are `userA@REALMONE` and `userB@REALMTWO`, respectively, map them to the login names `userA_REALMONE` and `userB_REALMTWO`, instead of `userA` or `userB`. This distinguishes the two users who belong to different realms.

MIT Kerberos client library support

Adaptive Server version 12.5.4 has been tested and is now certified with MIT client libraries using MIT Kerberos client libraries in MIT Kerberos version 1.4.x for UNIX platforms.

Adaptive Server version 12.5.4 also includes fixes that enhance the reliability of Adaptive Server with Kerberos under stress conditions.

LDAP user authentication enhancements

LDAP user authentication has been enhanced to include:

- Support for secondary server lookup
- LDAP user authentication robustness enhancements
- Failover in case of errors while communicating with the LDAP server

- Communication enhancements relasted to password expiration
- Tighter controls on login mapping

Configuring Adaptive Server for LDAP user authentication

LDAP user authentication allows client applications to send user name and password information to Adaptive Server for authentication by the LDAP server instead of syslogins. Authentication using the LDAP server allows you to use server-wide passwords instead of Adaptive Server or application-specific passwords. LDAP user authentication lets you simplify and centralize user administration.

LDAP user authentication works with directory servers that meet version 3 of the LDAP protocol standard, including Active Directory, iPlanet, and OpenLDAP Directory Server.

You can use two authentication algorithms with LDAP user authentication, which differ in how they obtain a user's distinguished name (DN). The algorithms use either:

- Composed DN for authentication, available for Adaptive Server version 12.5.1 or later, or,
- Searched DN for authentication, available for Adaptive Server version 12.5.2 and later.

The primary data structure used with the LDAP protocol is the LDAP URL.

An LDAP URL specifies a set of objects or values on an LDAP server. Adaptive Server uses LDAP URLs to specify an LDAP server and search criteria to use to authenticate login requests.

The LDAP URL uses this syntax:

```
ldapurl::=ldap://host:port/node/?attributes?base | one | sub?filter
```

Where:

- *host* – is the host name of the LDAP server.
- *port* – is the port number of the LDAP server.
- *node* – specifies the node in the object hierarchy at which to start the search.
- *attributes* – is a list of attributes to return in the result set. Each LDAP server may support a different list of attributes.

- base | one | sub – qualifies the search criteria. base specifies a search of the base node; one specifies a search of the base node and one sublevel below the base node; sub specifies a search of the base node and all node sublevels.
- filter – specifies the attribute or attributes to be authenticated. The filter can be simple, such as `uid=*`, or compound, such as `(uid=*)(ou=group)`.

Composed DN algorithm

The following steps describe the login sequence when you use the composed DN algorithm:

1 Open Client connects to an Adaptive Server listener port.

2 The Adaptive Server listener accepts the connection.

3 Open Client sends an internal login record.

4 Adaptive Server reads the login record.

5 Adaptive Server binds to the LDAP server with a DN composed from the primary URL and the login name from the login record. This bind also uses the password from the login record.

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

7 If the primary URL specifies a search, then Adaptive Server sends the search request to the LDAP server.

8 The LDAP server returns the results of the search.

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

Searched DN algorithm

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

1 Open Client connects to an Adaptive Server listener port.

2 The Adaptive Server listener accepts the connection.

3 Open Client sends an internal login record.

4 Adaptive Server reads the login record.

5 Adaptive Server binds to the LDAP server with a directory server access account.

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

   The connection established in steps 5 and 6 may persist between authentication attempts from Adaptive Server to reuse connections to DN searches.

7 Adaptive Server sends search requests to LDAP server based on the login name from the login record and the DN lookup URL.

8 The LDAP server returns the results of the search.

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

10 Adaptive Server uses the value of attribute as the DN and the password from the login record to bind to the LDAP server.

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

12 If the primary URL specifies a search, Adaptive Server sends the search request to the LDAP server.

13 The LDAP server returns the results of the search.

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

Adaptive Server reports a generic login failure to the client if any of these authentication criteria are not met.

You may skip steps 12 and 13 by not specifying search criteria in the primary or secondary URL strings. When you do not specify criteria in the primary or secondary URL strings, the authentication completes, displaying the success or failure returned by step 11.

Secondary lookup server support

Adaptive Server version 12.5.4 provides uninterrupted support to Adaptive Server clients authenticated by an LDAP server. You can now specify a secondary LDAP lookup server to fail over from a primary LDAP server in the event of the LDAP server failure or planned downtime.

The health of the URL set is monitored through the following states:

- INITIAL – indicates that LDAP user authentication is not configured.

- RESET – indicates that the URL has been entered with Adaptive Server administrative commands.
- READY – indicates that the URL is ready to accept connections.
- ACTIVE – indicates that the URL has performed a successful LDAP user authentication.
- FAILED – indicates that there is a problem connecting to the LDAP server.
- SUSPENDED – indicates that the URL is in maintenance mode, and will not be used.

The following sequence of events describe the failover and manual failback:

1 The primary and secondary URL sets are configured and in a READY state.

2 The connections are authenticated using the primary server infrastructure.

3 The primary server fails, and its state is changed to FAILED.

4 Connections automatically begin authentication through the secondary server infrastructure.

5 The primary server is repaired and brought back online by an LDAP administrator. The primary LDAP server state is changed by an Adaptive Server administrator to READY.

6 New connections are authenticated using the primary server.

Note Once Adaptive Server has failed over to the secondary LDAP server, a database administrator must manually activate the primary LDAP server before it can be used again.

When Adaptive Server encounters errors connecting to an LDAP server, it retries the authentication three times. If the errors persist the LDAP server is marked as FAILED. See “Troubleshooting LDAP user authentication errors” on page 357 for information on the LDAP errors which force Adaptive Server to get into a retry loop.

Adaptive Server version 12.5.4 introduces the following sp_ldapadmin options to support secondary lookup LDAP servers:

- To set the secondary DN lookup URL, enter:

```
sp_ldapadmin set_secondary_dn_lookup_url, <URL>
```

- To set the administrative access account for the secondary DN lookup URL, enter:

  ```
  sp_ldapadmin set_secondary_access_acct, <DN>,
  <password>
  ```

- To suspend the use of a primary or secondary URL for authentication, enter:

  ```
  sp_ldapadmin suspend, {primary | secondary}
  ```

- To activate the set of primary or secondary URLs for authentication, enter:

  ```
  sp_ldapadmin activate, {primary | secondary}
  ```

- To display details about the primary and secondary LDAP Server settings and status, enter:

  ```
  sp_ldapadmin list
  ```

  sp_ldapadmin list combines previous outputs from list_access_acct and list_urls. It has the following expected output for the primary and secondary servers:

  - Search URL
  - Distinguished Name Lookup URL
  - Access Account DN
  - Active [True | False]
  - Status [Ready | Active | Failed | Suspended | Reset]

Adaptive Server version 12.5.4 includes the following sp_ldapadmin option changes to support secondary servers.

- To display DN lookup URLs for the secondary server, enter:

  ```
  sp_ldapadmin list_urls
  ```

- To display the administrative account for the secondary DN lookup URL, enter:

  ```
  sp_ldapadmin list_access_acct
  ```

- To display new subcommands, enter:

  ```
  sp_ldapadmin [help | invalid sub-command]
  ```

LDAP server state transitions

The following tables list LDAP server state transitions when each of the sp_ldapadmin commands is executed.

Table 25-1 shows the state transitions when you execute sp_ldapadmin set_URL, where set_URL represents either of the following commands:

- set_dn_lookup_url
- set_primary_url
- set_secondary_dn_lookup_url
- set_secondary_url

Table 25-1: State transitions when sp_ldapadmin set_URL is executed

Initial state	Final state
INITIAL	RESET
RESET	RESET
READY	READY
ACTIVE	RESET
FAILED	RESET
SUSPENDED	RESET

Table 25-2 shows the state transitions when you execute sp_ldapadmin suspend.

Table 25-2: State transitions when sp_ldapadmin suspend is executed

Initial state	Final state
INITIAL	Error
RESET	SUSPENDED
READY	SUSPENDED
ACTIVE	SUSPENDED
FAILED	SUSPENDED
SUSPENDED	SUSPENDED

Table 25-3 shows the state transitions when you execute sp_ldapadmin activate.

Table 25-3: State transitions when sp_ldapadmin activate is executed

Initial state	Final state
INITIAL	Error
RESET	READY
READY	READY
ACTIVE	ACTIVE
FAILED	READY
SUSPENDED	READY

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

Table 25-4 shows the state transitions when Adaptive Server is restarted:

Table 25-4: State transitions when Adaptive Server is restarted

Initial state	Final state
INITIAL	INITIAL
RESET	RESET
READY	READY
ACTIVE	READY
FAILED	FAILED
SUSPENDED	SUSPENDED

Adaptive Server only attempts an LDAP login if the LDAP server is in a READY or ACTIVE state. Table 25-5 shows the state transitions:

Table 25-5: State transitions when an LDAP login succeeds

Initial state	Final state
READY	ACTIVE
ACTIVE	ACTIVE

Table 25-6 shows the state transitions when an LDAP login fails:

Table 25-6: State transitions when an LDAP login fails

Initial state	Final state
READY	FAILED
ACTIVE	FAILED

LDAP UA robustness enhancements

Adaptive Server version 12.5.4 introduces a number of new sp_ldapadmin options to enhance robustness.

Maximum ldapua native threads per engine

set_max_ldapua_native_threads sets the maximum number of native threads that can be running concurrently in an engine processing an LDAP authentication request.

```
sp_ldapadmin 'set_max_ldapua_native_threads', 'an integer'
```

The minimum value of set_max_ldapua_native_threads is 1. The maximum value is max native threads minus number of dump threads as specified using sp_configure. The default value is the same as the maximum value.

sp_configure ensures that max native threads is sufficient for set_max_ldapua_native_threads and the value of the configuration parameter number of dump threads.

LDAP request timeout

set_timeout sets the time in milliseconds that Adaptive Server waits for a response from the LDAP server before abandoning the authentication request.

You can set this option by entering:

```
sp_ldapadmin, 'set_timeout', 'time_in_milli_seconds'
```

The default value for set_timeout is 10,000 milliseconds (10 seconds.) Valid values are between 1 and 3,600,000 (one hour.)

Abandon LDAP authentication when full

set_abandon_ldapua_when_full allows you to seek alternative means of LDAP user authentication when the native threads per engine capacity is exceeded.

When no more threads are available, the request is abandoned if set_abandon_ldapua_when_full is set to true. If enable ldap user auth is set to 1, the client is authenticated using Adaptive Server syslogins. If enable ldap user auth is set to 2, the client login fails.

If set_abandon_ldapua_when_full is set to false, the authentication request is blocked until the LDAP descriptor can accept new authentication requests.

To set set_abandon_ldapua_when_full, enter:

```
sp_ldapadmin 'set_abandon_ldapua_when_full',
      'true | false'
```

The default value is false. Valid values are true and false.

LDAP descriptors per engine

The login sequence of searched DN algorithm requires Adaptive Server to bind to the LDAP server using the access account before it can perform searches. Adaptive Server obtains an LDAP descriptor (handle) as a result of the bind. This descriptor is used for searching the DN of the login on the LDAP server.

In Adaptive Server versions earlier than 12.5.4, there was only one descriptor per engine. While this descriptor was being used to perform a search by an incoming connection, other connections waited for the descriptor to become available. Adaptive Server version 12.5.4 can open up to 20 descriptors per engine. This provides improved concurrency and login performance.

For information about the searched DN algorithm, see “Searched DN algorithm” on page 350.

Troubleshooting LDAP user authentication errors

Adaptive Server may experience the following transient errors when communicating with the LDAP server. These errors are generally resolved by retrying the connection. If the errors persist after three retry attempts, Adaptive Server marks the LDAP server as FAILED.

- LDAP_BUSY – server is busy
- LDAP_CONNECT_ERROR – error during a connect
- LDAP_LOCAL_ERROR – error on the client side
- LDAP_NO_MEMORY – cannot allocate memory on the client side
- LDAP_OPERATIONS_ERROR – error on the server side
- LDAP_OTHER – unknown error code
- LDAP_ADMINLIMIT_EXCEEDED – a search has exceeded a limit
- LDAP_UNAVAILABLE – server cannot process the request
- LDAP_UNWILLING_TO_PERFORM – server is not going to process the request
- LDAP_LOOP_DETECT – a loop has been detected during a referral
- LDAP_SERVER_DOWN – server is not reachable (connection fails)

- LDAP_TIMEOUT – LDAP API fails because operation does not complete in the user-specified amount of time

Transient errors and a large number of simultaneous login requests could lead to a large number of repeated error messages in the error log. To increase the readability of the log, the following error message logging algorithm is used:

1 If a message is being logged for the first time, log it.

2 If the last time the message was logged was greater than 3 minutes:

   - Log the error message.
   - Log the number of times the message was repeated since the message was last printed.
   - Log the time elapsed, in minutes, since the message was printed.

Authentication failures arising from the following are not considered LDAP errors and are not conditions for retrying the authentication request.

- Bind failure due to bad password or an invalid distinguished name.
- A search after a successful bind that returns a result set of 0 or no attribute value.

Syntax errors found while parsing the URL are caught when an LDAP URL is set, and therefore do not fall into any of the above categories.

LDAP user authentication administration

The syntax of the sp_ldapadmin command is:

```
sp_ldapadmin command [, option1 [, option2]]
```

sp_ldapadmin command options include:

- `sp_ldapadmin 'set_primary_url', 'url'`
- `sp_ldapadmin 'set_secondary_url', 'url'`
- `sp_ldapadmin 'set_dn_lookup_url', 'url'`
- `sp_ldapadmin 'set_secondary_dn_lookup_url', 'url'`
- `sp_ldapadmin 'set_access_acct', 'distinguished name', 'password'`
- `sp_ldapadmin 'set_secondary_access_acct', 'distinguished name', 'password'`

- sp_ldapadmin 'suspend', {'primary' | 'secondary'}
- sp_ldapadmin 'activate', {'primary' | 'secondary'}
- sp_ldapadmin 'list'
- sp_ldapadmin 'list_urls'
- sp_ldapadmin 'list_access_acct'
- sp_ldapadmin 'check_url', 'url'
- sp_ldapadmin 'check_login', 'name'
- sp_ldapadmin 'set_timeout', timeout_in_milli_seconds
- sp_ldapadmin 'set_max_ldapua_native_threads', max_ldapua_native_threads
- sp_ldapadmin 'set_abandon_ldapua_when_full', {true|false}
- sp_ldapadmin 'help'

The following is an example output from a client administration session:

```
1> sp_configure 'enable ldap', 2
2> go

 Parameter Name                 Default     Memory Used Config Value
         Run Value   Unit                  Type
------------------------------ ----------- ----------- ------------
         ----------- --------------------- ----------
enable ldap user auth                     0           0            2
                   2 not applicable       dynamic

(1 row affected)
```

```
1> sp_ldapadmin
'set_primary_url','ldap://primldap:30001/'
2> go
The URL 'ldap://primldap:30001/' is set for LDAP User
Authentication.
(return status = 0)

1> sp_ldapadmin 'set_dn_lookup_url',
'ldap://primldap:30001/dc=sybase,dc=com??sub?uid=*'
2> go
The URL
'ldap://primldap:30001/dc=sybase,dc=com??sub?uid=*'
```

```
is set for LDAP User Authentication.
(return status = 0)

1> sp_ldapadmin
'set_access_acct','cn=directorymanager,dc=sybase,
dc=com', 'primpassword'
2> go
The LDAP account distinguished name 'cn=directory
manager,dc=sybase,dc=com' is set for LDAP user
authentication.
(return status = 0)

1> sp_ldapadmin 'set_secondary_url',
'ldap://secldap:31001/'
2> go
The URL 'ldap://secldap:31001/' is set for LDAP User
Authentication.
(return status = 0)

1> sp_ldapadmin 'set_secondary_dn_lookup_url',
'ldap://secldap:31001//dc=sybase,dc=com??sub?uid=*'
2> go
The URL
'ldap://secldap:31001//dc=sybase,dc=com??sub?uid=*'
is set for LDAP User Authentication.
(return status = 0)
2> sp_ldapadmin 'set_secondary_access_acct',
'cn=Manager,dc=sybase,dc=com', 'secpassword'
3> go
The LDAP account distinguished name
'cn=Manager,dc=sybase,dc=com' is set for LDAP user
authentication.
(return status = 0)
1> sp_ldapadmin activate, primary
2> go
(return status = 0)

1> sp_ldapadmin activate, secondary
2> go
(return status = 0)


1> sp_ldapadmin 'list'
2> go
Primary:
      URL:                 'ldap://primldap:30001/'
      DN Lookup URL:
```

```
      'ldap://primldap:30001/dc=sybase,dc=com??sub?uid
      =*'
      Access Account:
      'cn=directory manager,dc=sybase,dc=com
      Active:              'TRUE'
      Status:              'READY'
Secondary:
      URL:                 'ldap://secldap:31001/'
      DN Lookup URL:
     'ldap://secldap:31001/dc=syase,dc=com??sub?uid=*'
      Access Account:     'cn=Manager,dc=sybase,dc=com'
      Active:              'TRUE'
      Status:              'READY'
Timeout value:           '-1'(10000) milliseconds
Maximum LDAPUA native threads per Engine: '49'
Abandon LDAP user authentication when full: 'false'
(return status = 0)

1> sp_ldapadmin 'list_urls'
2> go
Primary URL:         'ldap://primldap:30001/'
Secondary URL:       'ldap://secldap:31001/'
Distinguished Name Lookup URL:
'ldap://primldap:30001/dc=sybase,dc=com??sub?uid=*'
Secondary Distinguished Name Lookup URL:
'ldap://secldap:31001/dc=sybase,dc=com??sub?uid=*'
(return status = 0)

1> sp_ldapadmin 'list_access_acct'
2> go
Access Account DN:
'cn=directory manager,dc=sybase,dc=com'
Secondary Access Account DN:
'cn=Manager,dc=sybase,dc=com'
(return status = 0)
```

LDAP user authentication tuning

You can configure and tune Adaptive Server options based on load of incoming connections and the Adaptive Server-LDAP server infrastructure. You can configure the following two options based on the number of simultaneous incoming requests:

- Use sp_configure to set max native threads, which indicates the number of native threads per engine.

- Use sp_ldapadmin to configure max_ldapua_native_threads, which indicates the number of LDAP user authentication native threads per engine.

Configure the following option based on the network and the health of the Adaptive Server/LDAP server infrastructure:

- Use sp_ldapadmin to configure set_timeout which indicates the LDAP server bind and search timeouts.

Configure the following option to specify Adaptive Server behavior when incoming connections have consumed max_ldapua_native_threads:

- Use sp_ldapadmin to configure set_abandon_ldapua_when_full.

LDAP user authentication password information changes

There are two LDAP user authentication-related informational messages that Adaptive Server obtains from the LDAP server and passes on to the client:

- If you log in to an Adaptive Server using an LDAP authentication mechanism with an LDAP user authentication password that is about to expire, the following message displays:

  ```
  Your password will expire in <number> days.
  ```

- If you attempt to log in to Adaptive Server using an LDAP authentication mechanism after the LDAP server administrator resets your password or after your LDAP server password has expired, you see a message 4002:

  ```
  Login failed
  ```

  If auditing is enabled and the errors auditing option is turned on, a 4099 message is sent to the audit log:

  ```
  sp_audit 'errors', 'all', 'all', 'on'
  ```

  The 4099 message reads:

  ```
  Your LDAP password has expired.
  ```

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

Adding tighter controls on login mapping

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

To map a user authenticated with Kerberos, use sybmapname instead of sp_maplogin. See instructions and examples in "Using sybmapname to handle user principal names" on page 344.

Only users with sso_role can create or modify login mappings using sp_maplogin.

In Adaptive Server version 12.5.4, controls have been added to sp_maplogin to avoid conflicts between an authentication mechanism setting for a login and a mapping that uses the login. Potential mapping conflicts are detected by the stored procedures sp_maplogin, sp_modifylogin, or sp_addlogin.

The tighter controls no longer permit a map:

- From one Adaptive Server login name to another login name
- From an external name that already exists as a local login
- To a nonexistent login name

Additionally, when the authentication mechanism is specified with a mapping, the mechanism is checked with the authentication mechanism set in the target login.

If a target login's authentication mechanism restricts the login to use a particular authentication mechanism, then the mechanism specified with the mapping must match either that specified for the login or match the "ANY" authentication mechanism.

When sp_maplogin detects that a conflict exists, sp_maplogin fails and reports an error to identify the conflict.

Similarly, sp_modifylogin and sp_addlogin check for an existing mapping that may conflict with the authenticate with option for the user login.

When sp_modifylogin or sp_addlogin detect a conflict, an error is reported to identify any conflicts with a login mapping.

Examples

Example 1 Maps an LDAP user to Adaptive Server "sa" login. A company has adopted LDAP as their repository for all user accounts and has a security policy that requires LDAP authentication of all users including database administrators, "adminA" and "adminB", who may manage hundreds of Adaptive Servers. Auditing is enabled and login events are recorded in the audit trail.

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

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

Use enable ldap user auth to require all users to authenticate using LDAP authentication:

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

When “adminA” authenticates during login to Adaptive Server, the distinguished name associated with “adminA” rather than only “sa” is recorded in the login audit event. This allows each individual performing an action to be identified in the audit trail.

Because “adminA” and “adminB” password is set in the LDAP server, there is no need to maintain the “sa” password on all Adaptive Servers being managed.

This example also allows different external identities and passwords to be used for authentication, while their actions within Adaptive Server still require the special privileges associated with “sa” account.

Example 2 Uses both PAM and LDAP to map users to application logins. A company has adopted both PAM and LDAP authentication but for different purposes. The company security policy defines LDAP as the authentication mechanism for general user accounts, and PAM for special users such as a middle-tier application. A middle-tier application may establish a pool of connections to Adaptive Server to handle requests on behalf of users of the middle-tier application.

Configure Adaptive Server for both LDAP and PAM user authentication:

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

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

```
sp_addlogin 'appX', password
go
sp_modifylogin appX, 'authenticate with', PAM
go
```

Instead of hard-coding a simple password in “appX” and maintaining the password consistently in several different Adaptive Servers, develope a custom PAM module to authenticate the application in a centralized repository using additional facts to verify the middle-tier application.

Client application login “appY” requires LDAP authentication of the user with its LDAP identity and password. Use sp_maplogin to map all LDAP authenticated users to login “appY”.

```
sp_addlogin 'appY', password
go
sp_maplogin LDAP, NULL, 'appY'
go
```

Users of “appY” are authenticated with their company identity and password, then mapped to a local Adaptive Server login “appY” to execute database actions. Authentication has occurred with the identity of the LDAP user, which is recorded in the audit trail, and executes with permissions appropriate to the application login “appY”.

SSL support

Adaptive Server version now supports SSL on:

- HP-IA64
- Linux64

PAM support

Adaptive Server version 12.5.4 supports PAM on:

- HP-IA64
- Macintosh OS X

Updates to encrypted columns

Adaptive Server version 12.5.4 includes the following encrypted columns enhancements. For information about the first version of the encrypted columns feature which was part of version 12.5.3a, see *Using Encrypted Columns in Adaptive Server*.

Referential integrity with encrypted columns

You can establish referential integrity between two encrypted columns when:

- You use the same key to encrypt the referenced and referencing columns.
- The key used to encrypt the columns specifies an initialization vector NULL and a random pad NULL.

Referential checks are efficient as they are performed on encrypted values.

alter table and encrypted columns

You cannot use alter table to encrypt or decrypt a column belonging to a clustered or placement index. To encrypt or decrypt this column, drop the index, alter the column, and re-create the index.

You cannot use alter table to decrypt a column if the table has a trigger defined. To decrypt the column, drop the trigger, alter the column, then re-create the trigger.

sp_help and encrypted columns

sp_help displays information about encryption keys. When a key name is specified as the parameter to sp_help, the command lists the key's name, owner, object type, and creation date.

sp_helprotect and encrypted columns

sp_helprotect reports new information on encrypted columns, encryption keys, and users as follows:

- Tables and columns – reports who has been granted decrypt permission and on which columns.
- Encryption keys – reports who has been granted select permission.
- Users – indicates users who have been granted create encryption key permission.

Password complexity and login options

Adaptive Server version 12.5.4 combines new and old mechanisms that let you to establish rules about passwords for new logins or for passwords that are being reset.

As with earlier versions of Adaptive Server, you can:

- Require that passwords contain at least one digit.
- Require that passwords have a minimum length.
- Set a password expiration period.
- Require that a login is locked out after a certain number of failed attempts.
- Set an individual login to have its own rules for digits, minimum length, and login failures. The per-login rules override the global logins for that user.

However, in Adaptive Server version 12.5.4 you can also:

- Specify that a login name cannot be a substring of the password.
- Set a minimum number of special characters for the password.
- Set a minimum number of alphabetic characters for the password
- Set a minimum number of upper-case letters for the password.
- Set a minimum number of lower-case letters for the password.
- Specify that the password must be reset is the first time a login is used.
- Set a minimum number of digits for the password.
- Set a password expiration warning interval.

You can set each of these options in the Adaptive Server plug-in, or you can use:

```
sp_passwordpolicy 'set', option, value
```

For information about each new option and its valid values, see “New password complexity checks,” below.

Setting password complexity options creates a row for each option in the sysattributes table. If a row exists for a new option, precedence checking uses the new option value and ignores any older corresponding option values.

To return to earlier versions of password rules, unselect the password complexity options either, using the Adaptive Server plug-in, or use:

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

The new password complexity options include cross checks. For example, if the sum of the min lower case in password and min upper case in password is greater than the min alpha in password, a warning message displays.

New password complexity checks

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

To turn off an option for an individual option, enter:

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

To turn off password policy options for all passwords, enter:

```
sp_passwordpolicy 'clear'
```

Disallowing simple passwords

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

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

To set this option, enter:

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

When you disallow simple passwords, you cannot use your login name as a substring in your password. You must set it to something complex. For example:

```
sp_password 'old_complex_password', BHotAcha789, johnd
```

The login `johnd` now has a password of `BHotAcha789`, which does not contain the login name as a substring.

However, if you change the login password entering the following, the login `johnd` is now a substring of the new password `johnd123`, and the command fails:

```
sp_password 'old_complex_password', johnd123, johnd
```

Custom password-complexity checks

Adaptive Server version 12.5.4 allows you to custom-configure password checking rules using two stored procedures:

- sp_extrapwdchecks
- sp_cleanpwdchecks

These stored procedures are defined and located in the master database and are automatically invoked during Adaptive Server password complexity checks, and when dropping a login, respectively. See “Enabling custom password checks” on page 379 for an example of how to create these custom stored procedures.

Specifying a minimum number of digits in a password

Use min digits in password to specify the minimum number of digits in a password. Valid values are:

- 0 through 16 – the minimum number of digits that must exist in a password.
- -1 – the password cannot contain digits.

By default, this password complexity option is turned off and this check is not applied to passwords.

To set this option, enter:

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

For example, if you have set min digits in password to 4, you must have at least four digits in your password.

Specifying a minimum number of alphabetic characters in a password

min alpha in password specifies the minimum number of alphabetic characters allowed in a password. This value must be at least the sum of minimum number of uppercase characters and minimum number of lowercase characters.

Valid values are:

- 0 through 16 – the number of alphabetic characters required to be in the password.
- -1 – the password cannot contain alphabetic characters.

By default, this password complexity option is turned off and this check is not applied to passwords.

To set the minimum number of alphabetic characters in a password, enter:

```
sp_passwordpolicy 'set', 'min alpha in password',
      number
```

For example, if you have set min alpha in password to 4, you must have at least 4 alphabetic characters in your password.

```
sp_addlogin 'johnd', 'sec123456'
```

Specifying a minimum number of special characters in a password

min special char in password specifies the minimum number of special characters for a password. Valid values are:

- 0 through 16 – the minimum number of special characters required for a password.
- -1 – the password cannot contain special characters.

By default, this password complexity option is turned off and this check is not applied to passwords.

To set the minimum number of special characters in a password, enter:

```
sp_passwordpolicy 'set', 'min special char in
      password',number
```

Specifying a minimum number of uppercase letters in a password

min upper char in password allows you to set the minimum number of uppercase letters for a password. Valid values are:

- 0 through 16 – the number of uppercase letters required for a password.

- -1 – the password cannot contain uppercase characters.

By default, this password complexity option is turned off and this check is not applied to passwords.

To set the minimum number of uppercase characters allowed in a password, enter:

```
sp_passwordpolicy 'set', 'min upper char in password'' 
      number
```

Specifying a minimum number of lowercase letters in a password

min lower char in password sets the minimum number of lowercase letters for a password. Valid values are:

- 0 through 16 – indicates the number of lowercase letters required for a password.
- -1 – indicates that the password cannot contain lowercase characters.

By default, this password complexity option is turned off and this check is not applied to passwords.

To set the minimum number of lowercase letters in a password, enter:

```
sp_passwordpolicy 'set', 'min lower char in password', 
      number
```

For example, if you set min lower char in password to 3, you can create a login `johnd` with the password `abcdEF1#`, by entering:

```
sp_addlogin 'johnd', 'abcdEF1#'
```

However, if you attempt to add the following login and the minimum number of lowercase letters allowed is 3, the command fails:

```
sp_addlogin 'johnd', 'abCDE#1'
```

Specifying the minimum password length

minimum password length sets the minimum password length. You can set a minimum password length from 0 to 30. The value you specify with must be at least the sum of all other minimum requirements.

For example, minimum password length must be set to at least 10 if you have set:

- minimum digits in password to 3
- minimum special characters in password to 2

- minimum uppercase characters in password to 2
- minimum lowercase characters in password to 3

In this example, if the password length is less than 10, a warning message displays, but the setting of the password policy option succeeds.

To set the minimum password length, enter:

```
sp_passwordpolicy 'set', 'minimum password length',
      number

sp_password 'old_complex_password', 'joh12', 'johnd'
```

Specifying the password expiration

password expiration specifies the number of days a password can exist before it expires. You specify this value on a global basis. Valid values include:

- 0 – the password will never expire.
- 1 through 32767 – the number of days the password can exist without expiring.

By default, this password complexity option is turned off and this check is not applied to passwords.

To specify the password expiration date, enter:

```
sp_passwordpolicy 'set', 'password expiration', number
```

Specifying the password expiration warning interval

password exp warn interval indicates the number of days before a password expires that the password expiration warning messages displays. These messages display with every successful login until the password is changed or it expires. This value must be less than or equal to the password expiration.

Valid values are 0 to 365. This option is turned off by default.

To specify the password expiration warning interval, enter:

```
sp_passwordpolicy 'set', 'password exp warn interval',
      number
```

Specifying the number of failed logins allowed

maximum failed logins specifies the maximum number of failed logins that can occur before the login is locked. You specify this value globally. Valid values are:

- 0 – logins are never locked, regardless of the number of failed login attempts.
- 1 through 32767 – the number of failed logins that can occur before the login is locked.

By default, this value is turned off and this check is not applied to passwords.

To set the number of failed logins allowed, enter:

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

Resetting the password at first login

expire login changes the login status to expired when a System Security Officer creates or resets a login. The login is then required to change the password on the first login. Valid values are:

- 0 – new or reset logins will not expire.
- 1 – new or reset logins expire; you must reset your password at the first login.

By default, this value is turned off and this check is not applied to passwords.

To require a change of password at first login, enter:

```
sp_password policy 'set', 'expire login', [ 1 | 0 ]
```

Password complexity option cross-checks

Some password complexity options have interaction implications:

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

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

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

Setting old and new password complexity checks

Table 25-7: Old and new password complexity checks

Password checks and policies for Adaptive Server authentication	Existing configuration parameters specified using sp_configure	New password complexity options specified using sp_passwordpolicy	Existing per-login overrides specified using sp_modifylogin
Password expiration	system-wide password expiration	system-wide password expiration	password expiration
Digits in password	check password for digit	min digits in password	N/A
Alphabetic characters in password	N/A	min alpha in password	N/A
Password length	minimum password length	minimum password length	min passwd length
Failed logins lockout	maximum failed logins	maximum failed logins	max failed_logins
Disallow simple passwords	N/A	disallow simple passwords	N/A
Special characters in password	N/A	min special char in password	N/A
Uppercase letters in password	N/A	min upper char in password	N/A
Lowercase letters in password	N/A	min lower char in password	N/A
Password expiration warning interval	N/A	password exp warn interval	N/A
Resetting your password at first login	N/A	expire login	N/A
Custom password complexity checks	N/A	N/A	N/A

You can set the password complexity options at:

- The login level using sp_addlogin or sp_modifylogin.
- The global level using the new sp_passwordpolicy or sp_configure.

Because you can set password configuration options on a global and per-login basis, and using old and new parameters, the order of precedence in which the password options will be applied is important

When applying password options, the order of precedence is:

1 Existing per-login parameters

2 New password complexity options

3 Existing global password options

Examples

Example 1 If you enter the following, you have set the minimum password length for "johnd" to 6:

```
sp_addlogin @login_name = 'johnd',
      @passwd = 'complex_password',
      @minpwdlen = 6
```

If you then enter the following existing global options for login "johnd", you have created two different minimum password length requirements for login "johnd", and you have also set restrictions about digits in the password:

```
sp_configure 'minimum password length', 8
sp_configure 'check password for digit', 'true'
sp_passwordpolicy 'set', 'min digits in password', 2
```

If you then try to create a password for login "johnd" as follows:

```
sp_password @caller_password = 'old_complex_password',
@new_password = 'abcd123', @login_name = 'johnd'
```

Adaptive Server checks the password in the following order:

1 Per-login existing options check: minimum password length must be greater than 6. This is true and the check passes.

2 New options: minimum digits in password must be greater than 2. This is true and the check passes.

3 Existing global options: minimum password length specified here is not checked because there is already a per-login check for the login "johnd".

4 The check password for digit option is redundant because it is already checked when the minimum number of digits is turned on and set to 2.

Once these checks have been performed in the designated sequence, and the new password for login "johnd" passes these checks, the new password is successfully created.

Example 2 If, for the same login, you enter the following, Adaptive Server first checks the per-login existing options, and determines the minimum password length is set to 6, but that you have attempted to create a password with only 4 characters:

```
sp_password @caller_password = 'old_complex_password',
@new_password = 'abcd', @login_name = 'johnd'
```

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

Example 3 If you attempt to create a new login with the following password configuration options, this sets the minimum password length for login `johnd` to 4:

```
sp_addlogin @login_name = 'johnd', @passwd =
'complex_password', @minpwdlen = 4
```

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

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

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

```
sp_password @caller_password = 'old_complex_password',
@ new_password = 'abcde', @login_name = 'johnd'
```

Adaptive Server performs the checks in the following order:

1 Per-login existing options check: the minimum password length of a new password is 4. The password “abcde” is greater than 4, so this check passes.

2 New global requirement check: the minimum digits in a password is set to 1, globally. This check fails.

Adaptive Server does not create a new password and prints an error message.

To create a new password, all the checks must pass.

Stored procedures for password complexity

Adaptive Server includes several stored procedures to help configure password complexity.

sp_extrapwdchecks

sp_extrapwdchecks is a custom stored procedure that can contain user-defined logic for password complexity checks. You can configure sp_extrapwdchecks according to your security needs. Install sp_extrapwdchecks in the master database.

```
sp_extrapwdchecks caller_password, new_password, login_name
```

Where:

- *caller_password* – specifies the current password.
- *new_password* – specifies the new password being set.
- *login_name* – specifies the login name associated with the password being changed or added.

sp_extrapasswordchecks must use raiserror to signal a failure to Adaptive Server. Use sp_addmessage to add error message for this failure in Adaptive Server.

sp_cleanpwdchecks

sp_cleanpwdchecks is a custom stored procedure that allows you to define when and how to remove login and password-related attributes stored in user-defined tables. sp_cleanpwdchecks is user-defined, and is dynamically called in the master database when you drop a login.

```
sp_cleanpwdchecks, login_name
```

Where:

- *login_name* – specifies the login name of the cleanup to be performed.

sp_passwordpolicy

sp_passwordpolicy is an interface that a user with sso_role can use to specify, remove, and list new password complexity options. This information is stored in the master.dbo.sysattributes table.

```
sp_passwordpolicy {set | clear | list }, policy_option, option_value
```

where:

- set – sets a value to an option.
- clear – deletes the row for the option specified in the master.dbo.sysattributes table. If there is no policy option specified, clear deletes all the option rows in the sysattributes table.

- list – lists the values of the options specified.
- *policy_option* – the option parameter on which to perform an operation. Valid options are:
  - disallow simple passwords – a value of 1 turns this option on, and a value of 0 turns it off.
  - min digits in password – indicates the minimum number of digits to be allowed in a password.
  - min alpha in password – indicates the minimum number of alphabetic characters in a password.
  - min special char in password – indicates the minimum number of special characters allowed in a password.
  - min upper char in password – indicates the minimum number of upper-case characters allowed in a password
  - min lower char in password – indicates the minimum number of lower case characters allowed in a password.
  - systemwide password expiration – indicates the system-wide password expiration in days.
  - password exp warn interval – indicates the password expiration warning interval in days.
  - minimum password length – sets the minimum length of the password.
  - maximum failed logins – sets the maximum number of failed logins allowed in a session before the account is locked.
  - expire login – specifies that a login status changes to expired status when you create or reset your login. You are required to change your password on your first login.
- *option_value* – is the value for the *policy_option*.

Examples

Example 1 Sets a password expiration warning interval to seven days before the password expires:

```
sp_passwordpolicy 'set',
      'password exp warn interval', 7
```

Example 2 Lists the option for minimum number of special characters:

```
sp_passwordpolicy 'list',
      'min special char in password'
```

Example 3 Resets disallow simple passwords to the default value:

```
sp_passwordpolicy 'clear', 'disallow simple passwords'
```

Changed stored procedures

The following stored procedures have been modified for password complexity implementation:

- sp_addlogin and sp_password – invoke the new password complexity checks.
- sp_droplogin – if sp_cleanpwdchecks is present in the master database, is executed after dropping a login with sp_droplogin.
- sp_displaylogin – new sp_passwordpolicy security options are taken into consideration when displaying login information related to password expiration, maximum failed logins, and password length.

Enabling custom password checks

Adaptive Server version 12.5.4 allows a system security officer to write user-defined stored procedures that enable custom password checks.

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

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

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

```
create proc sp_extrapwdchecks
(
@caller_password varchar(30), --the current password of caller
@new_password    varchar(30), -- the new password of the target acct
@loginame        varchar(30), -- user to change password on
)
as

begin
```

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


select @changedby = suser_name()

-- Change this line according to your installation.
-- This keeps history of 12 months only.
select @current_time = getdate(),
       @cutoffdate = dateadd(month,-12,getdate())
select @encrypted_pwd = internal_encrypt(@new_password)

delete master..pwdhistory
    where  name = @loginame
    and    pwdate < @cutoffdate

if not exists ( select 1 from master..pwdhistory
                where name = @loginame
                and   password = @encrypted_pwd )
begin
      insert master..pwdhistory
      select @loginame, internal_encrypt(@caller_password),
             @current_time, @changedby
      return (0)
end
else
begin
      raiserror 22001  --user defined error message
      return (1)
end
end
go
```

Use sp_addmessage to add the user-defined message 22001. A raiserror 22001 indicates a custom password-complexity check error and leads to a failure of sp_addlogin or sp_password.

The following user-defined stored procedure can be used for clean-up purposes after adding history using sp_extrapwdchecks.

```
create proc sp_cleanpwdchecks
(
            @loginame      varchar(30)
                        -- user to change password on
)
```

```
as
begin

delete master..pwdhistory
where name = @loginame
end
      go
```

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

DDLGen support

Logins – TL generates DDL for one or all logins. This example generates DDL for all logins on a machine named HARBOR using port 1955:

```
ddlgen -Uroy -Proy123 -SHARBOR:1955 -TL -N%
```

Note The password in the DDL generated for all logins is "password".

Alternatively, you can specify an individual login by using -N*username* instead of -N%:

```
ddlgen -Ulogin -Ppassword -Sserver:port -TL -Nusername
```

If server-wide password complexity options have been specified for the login or logins, all sp_addlogin and sp_modifylogin DDL statements are generated first, followed by DDL statements for the password-complexity options.

This example generates DDL for the login "george" on a machine named HARBOR using port 1955:

```
ddlgen -Uroy -Proy123 -SHARBOR:1955 -TL -Ngeorge
```

Adaptive Server plug-in support

The Adaptive Server plug-in supports GUI administration for password complexity.

Exporting set options from a login trigger

Adaptive Server version 12.5.4 enables options set inside login triggers to remain valid for the entire user session.

The following set options are automatically exported:

- showplan
- arithabort [overflow | numeric_truncation]
- arithignore [overflow]
- colnames
- format
- statistics io
- procid
- rowcount
- altnames
- nocount
- quoted_identifier
- forceplan
- fmtonly
- close on endtran
- fipsflagger
- self_recursion
- ansinull
- dup_in_subquery
- or_strategy
- flushmessage
- ansi_permissions
- string_rtruncation
- prefetch
- triggers
- replication

- sort_resources
- transactional_rpc
- cis_rpc_handling
- strict_dtm_enforcement
- raw_object_serialization
- textptr_parameters
- sort_merge
- remote_indexes
- explicit_transaction_required
- statement_cache
- command_status_reporting
- proc_return_status
- proc_output_params

Setting global login triggers

Adaptive Server version 12.5.4 includes a new global login trigger. Use sp_logintrigger to set a global login trigger, that is executed at each user login. To take user-specific actions, set a user specific login trigger using sp_modifylogin or sp_addlogin.

Note You can activate this option by setting trace flag -T4073.

sp_logintrigger

Description

sp_logintrigger can be used to set and display the global login trigger. This global login trigger has the same characteristics as a personal login script. It is executed before any personal login script for every user that tries to log in, including system administrators and security officers.

Syntax

sp_logintrigger <*global login trigger name*>

Parameters

global login trigger name
is the name of the global login trigger.

If no parameter is included, sp_logintrigger displays the current login trigger status and name if it exists, and no rows if there is no global login trigger defined.

Examples

Example 1 To set a global login trigger using sp_logintrigger:

```
sp_logintrigger 'master.dbo.myproc'
```

Example 2 To view an updated global login trigger:

```
1> sp_logintrigger
2> go
Global login trigger               Status
---------------------------------- -------
sybsystemprocs.dbo.myproc          Enabled

(1 row affected)
(return status = 0
```

Example 3 When a global login trigger does not exist:

```
1> sp_logintrigger
2> go
Global login trigger Status
-------------------- ------
(0 rows affected)
```

Example 4 To delete a global login trigger that was specified earlier with sp_logintrigger, enter:

```
sp_logintrigger 'drop'
```

Usage

- Global variable *@@logintrigger* to find out if a global login trigger is defined and enabled.

- There is a difference between this global login and the private login script. This global login trigger is stored by name in sysattributes, while the private login script is stored only by object ID.

Permissions

Any user can execute sp_logintrigger to display the current global login trigger. To set a new login trigger, sso_role is required.

CHAPTER 26

Miscellaneous New Features

This chapter includes descriptions for these features included in Adaptive Server version 12.5.4.

Adaptive Server plug-in support

In Adaptive Server version 12.5.4, the Sybase Central™ Adaptive Server plug-in supports archive database access and configuring password complexity options.

Use the plug-in to manage archive databases. There is a new folder called Archive Databases under the Databases folder.

For information on using the Adaptive Server plug-in to manage archive database access, see Chapter 19, “Controlling Access to Adaptive Server” in *Managing Adaptive Server Enterprise*.

Use the plug-in to configure Adaptive Server for password complexity options from the Server Property Sheet. There is a new tab in the Server Property Sheet called Login Password Configuration. For information on using the Adaptive Server plug-in to configure password complexity options, see Chapter 7, “Managing Databases” in *Managing Adaptive Server Enterpise*.

Changed monitoring tables

Table 26-1 describes the changes that have been made to monitoring tables in Adaptive Server version 12.5.4.

Table 26-1: Updates to monitoring tables

Monitoring table	Column name	Description
monSysStatement monProcessStatement	RowsAffected	Indicates the number of rows affected by the current statement. May be helpful when you are looking for queries using an inefficient query plan, because these queries probably show a high number of logical I/Os per returned row.

Enhancements on Linux platforms

Large memory support

Large memory support allows you to set the extended cache size configuration parameter to a value other than 0.

In versions of Adaptive Server earlier than 12.5.4, the wash size of the smallest I/O memory pool in the primary data caches was set to a fixed value. Any configured value was overridden.

Adaptive Server version 12.5.4 allows you to configure the wash size for for the smallest I/O memory pool in the primary data cache.

DIRECTIO support

Adaptive Server version 12.5.4 does not does not support DIRECTIO on file system devices.

Large memory support and POSIX Async I/O

Adaptive Server version 12.5.4 supports the large memory support feature on Linux platforms configured to use POSIX Async I/O.

JRE support

Adaptive Server version 12.5.4 includes JRE 1.4, which is installed by default in full and typical installations, and in custom installations whenever a component that requires the JRE is selected for installation.

Sybase driver support

Adaptive Server version 12.5.4 contains ODBC and OLE DB drivers developed by Sybase. The third-party rebranded ODBC and OLE DB Driver Kits included with earlier versions are no longer shipped with Adaptive Server. The discontinued drivers continue to be supported as per the schedule described below.

The retired ODBC Driver Kit was installed in *%SYBASE%\ODBC*, and registered with the ODBC Driver Manager as "Sybase ASE ODBC Driver". The new ASE ODBC Driver by Sybase is installed in *%SYBASE%\DataAccess\ODBC*, and registered as "Adaptive Server Enterprise". The version shipping with Adaptive Server version 12.5.4 is version 12.5.1.510.

The retired OLE DB Driver Kit was installed in *%SYBASE%\OLEDB*, and used the provider short name of "Sybase.ASEOLEDBProvider" and the long name of "Sybase ASE OLE DB Provider". The new ASE OLE DB Provider by Sybase is installed in *%SYBASE%\DataAccess\OLEDB*, and uses provider short name "ASEOLEDB." The version shipping with Adaptive Server 12.5.4 is version 12.5.1.510.

Ongoing support for third-party drivers

Existing customers with valid support contracts who are already using the third-party ODBC Driver and OLE DB Provider can continue to receive support for these products through the life cycle of Adaptive Server 12.5.

Until July 30, 2007, the support program will continue as it is today. After July 30, 2007, Sybase will address only P1 (severity level 1) issues. New updates to these drivers will be delivered as a standalone downloads.

The third-party ODBC Driver and OLE DB Provider you have today will support Adaptive Server features up to and including Adaptive Server 12.5.3. Any issues found with these drivers will only be addressed by Sybase if they can be reproduced in version 12.5.3. If you use the third-party products with versions of Adaptive Server later than 12.5.3, Sybase may not be able to identify problems or support fixes.

See the Sybase Web site at http://www.sybase.com/detail?id=1040652 for the End of Life notification for these drivers.

Sybase recommends that you migrate to the new ODBC and OLE DB drivers as soon as possible.

See the *New Features for OpenServer 12.5.1 and SDK 12.5.1 for Windows, Linux and UNIX*, for instructions on migrating to the new drivers.

CHAPTER 27

Archive Database Access

Adaptive Server version 12.5.4 introduced the archive database access feature.

Archive database access documentation

For complete archive database access documentation, see Chapter 4, “Archive Database Access.”

CHAPTER 28

Shared Directory Changes

Adaptive Server version 12.5.4 has made the following changes to its shared directory structure.

Shared directory changes

Adaptive Server version 12.5.4 includes a number of changes to the shared directory structure. Table 1 shows the directory

Table 1: Shared directory structure changes for UNIX and Linux platforms

Component	Old location	New location
Shared directory	*$SYBASE/shared-1_0*	*$SYBASE/shared*
Sybase Central	*$SYBASE/sybcent41*	*$SYBASE/shared/sybcentral43*
JRE	*$SYBASE/shared-1_0/JRE-1_3*	*$SYBASE/shared/jre142*
Shared JAR file	*$SYBASE/shared-1_0/lib*	*$SYBASE/shared/lib*

Table 2: Shared directory structure changes for Microsoft Windows platforms

Component	Old location	New location
Shared directory	*%SYBASE%\shared-1_0*	*%SYBASE%\Shared*
Sybase Central	*%SYBASE%\sybcent41*	*%SYBASE%\Shared\Sybase Central 4.3*
JRE	*%SYBASE%\shared-1_0\JRE-1_3*	*%SYBASE%\Shared\Sun\jre142*
Shared JAR file	*%SYBASE%\shared-1_0\lib*	*%SYBASE%\Shared\lib*

Note All Adaptive Server components have been modified to use the new directory structure.

Impacted directories

Adaptive Server contains the following Java applications that are affected by this change:

- Sybase Central
- ASE Plug-in
- DDLGen
- ASE Replicator
- Web Services Producer and Consumer
- SQL Debugger
- Migration Tool
- Job Scheduler

CHAPTER 29

Changes to Stored Procedures, Functions, and Commands

This chapter documents changes in Adaptive Server version 12.5.4 not documented elsewhere.

New syntax for *shutdown*

Adaptive Server version 12.5.4 includes the following new syntax for shutdown:

```
shutdown [srvname] [with {wait [="hh:mm:ss"] | nowait}]]
```

Parameters

srvname

is the logical name by which the Backup Server is known in the sysservers system table. This parameter is not required when shutting down the local Adaptive Server.

with wait

gracefully shuts down the Adaptive Server or Backup Server with wait is the default.

hh:mm:ss

is an optional setting that specifies the maximum time the server waits for all running or sleeping processes to finish.

with nowait

shuts down the Adaptive Server or Backup Server immediately, without waiting for currently executing statements to finish.

Note Using shutdown with nowait can lead to gaps in IDENTITY column values.

Specifying a wait time

When the server prepares to shut down, it:

1 Performs checkpoint on all the databases.

2 Prevents any new user from logging in.

3 Waits for all running or sleeping processes to finish their job.

4 Performs another checkpoint on the databases, this time with a flag indicating it needs to flush:

- All dynamic thresholds in mixed log-data databases.
- All object statistics.
- The values of the identity fields to avoid holes after recovery.

When you use with wait with the *hh:mm:ss option*, the time you specify is not the maximum total time Adaptive Server takes to shut down. Instead, Adaptive Server takes into account the time it takes to perform the first checkpoint, and automatically subtracts this from the time you specified.

For example, if you specify a maximum wait time of 20 minutes and the first checkpoint takes 3 minutes, Adaptive Server allows up to 17 minutes for processes to finish. If, for some reason the second checkpoint takes longer, however, this is not calculated into the with wait *hh:mm:ss* parameter you specify.

Adaptive Server also allows a checkpoint to take longer than the time you specify in with wait *hh:mm:ss*. For example, if you specify a wait time of 10 minutes but the first checkpoint takes 20 minutes to complete, Adaptive Server does not interrupt the checkpoint, but waits for it to complete. Adaptive Server immediately begins to shut down once checkpoint is complete, since the time you specified has passed, and runs the last checkpoint with the flag informing you of the flushes you must perform.

Expanded *select* * syntax

When the source text of a stored procedure or trigger is stored in the system table syscomments, a query using select * is stored in syscomments expanding the column list referenced in the select *.

For example, a select * from a table containing the columns col1 and col2 is stored as:

```
select <table>.col1, <table>.col2 from <table>
```

In Adaptive Server version 12.5.4, the expanding of the column-list has been enhanced to check whether identifiers (table-names, column-names and so on) comply with the rules for identifiers.

For example, if a table includes the columns col1 and 2col, the second column-name starts with a number, which can only be included by using brackets in the create table statement.

When performing a select * in a stored procedure or trigger from this table, the text in syscomments looks similar to:

```
select <table>.col1, <table>[2col] from <table>
```

For all identifiers used in the text that exands a select *, brackets are added when the identifier does not comply with the rules for identifiers.

You must add brackets around identifiers to make sure Adaptive Server can use the SQL-text while performing an upgrade to a more recent release.

dump database and *load database* with verification

In Adaptive Server 12.5.4, the dump database and load database commands introduce an option that allows you to verify database dumps and loads.

The syntax for dump database is:

```
dump database <dbname> with verify [ = header | full ]
```

The syntax for load database is:

```
load database <dbname> with verify[only] [= header |
full ]
```

When you execute dump database, Backup Server performs minimal header and structural row checks for data pages as they are being copied to the archives. There are no structural checks done at this time to global allocation map (GAM) and object allocation map (OAM) pages, indexes, text, or log pages.

You can perform the same checks when loading an archive using the command load with verify[only].

An archive can be checked without a physical load, using the load database with verifyonly command: this option displays the dump header information similar to the load option of with headeronly.

Allowing updates to system catalogs

The server-wide configuration option allow updates to system catalogs takes precedence over the stored procedure settings for allow updates. If you do not enable allow updates to system catalogs at the server level, individual, stored procedure settings determine whether you can modify system catalogs.

Modulo arithmetic for numeric datatypes

In Adaptive Server version 12.5.4, you can perform modulo arithmetic on real, float, decimal, and numeric datatypes, as well as on integers.

IPv6 IP address requirements

IPv6 architecture an IP address length of 64 bytes. Adaptive Server version 12.5.4 introduces a new function to return information from the process status structure (pss):

```
pssinfo(<spid | 0>, '<pss field>')
```

where:

- *spid* – process ID. When you enter 0, the current process is used.

- *pss field* – valid values are:
  - ipaddr – client IP address.
  - extusername – when using external authentication like (PAM, LDAP), extusername returns the external PAM or LDAP user name used.
  - dn – distinguished name when using LDAP authentication.

  The pssinfo function also includes the option to display the external user name and the distinguish name.

External transactions

Adaptive Server version 12.5.4 includes two functions you can use to decode external transactions, xa_bqual and xa_gtrid.

xa_bqual

Description — Returns the binary version of the bqual component of an ASCII XA transaction ID.

Syntax — xa_bqual(*xid*, 0)

Parameters

xid
is the ID of an Adaptive Server transaction, obtained from the xactname column in systransactions or from sp_transactions.

0
is reserved for future use.

Examples — **Example 1** Returns "0x227f06ca80", the binary translation of the branch qualifier for the Adaptive Server transaction ID "0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0". The Adaptive Server transaction ID is first obtained using sp_transactions:

```
1> sp_transactions

xactkey                        type     coordinator starttime           st
ate      connection dbid  spid  loid  failover    srvname  namelen  xactna
me
```

```
------------------------------ -------- ----------- ------------------- --
-------- ---------- ----- ----- ----- ----------- -------  -------  ------
---------------------------------
0x5316000006000000017e4885b0700 External XA          Dec 9 2005  5:15PM In
Command  Attached     7   20    877  Resident Tx  NULL         39
0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0

1> select xa_bqual("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go

...

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

0x227f06ca80
```

Example 2 xa_bqual is often used with xa_gtrid. This example returns the global transaction IDs and branch qualifiers from all rows in systransactions where its coordinator column is the value of "3":

```
1> select gtrid=xa_gtrid(xactname,0),
    bqual=xa_bqual(xactname,0)
    from systransactions where coordinator = 3
2> go

        gtrid

        bqual

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

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

         0xb1946cdc52464a61cba42fe4e0f5232b

         0x227f06ca80
```

Usage

If an external transaction is blocked on Adaptive Server and you are using sp_lock and sp_transactions to identify the blocking transaction, you can use the XA transaction manager to terminate the global transaction. However, the value of xactname that sp_transactions returns is in ASCII string format, while XA Server uses an undecoded binary value. Use xa_bqual to determine the bqual portion of the transaction name in a format recognized by the XA transaction manager.

xa_bqual returns:

- The translated version of the string that follows the second "_" (underscore) and precedes either the third "_" or end-of-string value, whichever comes first.
- NULL if the transaction ID cannot be decoded, or is in an unexpected format.

Note xa_bqual does not perform a validation check on the xid.

Standards — ANSI SQL – Compliance level: Transact-SQL extension.

Permissions — Any user can use xa_bqual.

See also — **Function** xa_gtrid

Stored procedures sp_lock, sp_transactions

xa_gtrid

Description — Returns the binary version of the gtrid component of an ASCII XA transaction ID.

Syntax — xa_gtrid(*xactname*, *int*)

Parameters — *xid*
is the ID of an Adaptive Server transaction, obtained from the xactname column in systransactions or from sp_transactions.

0
is reserved for future use.

Examples — In this typical situation, returns "0x227f06ca80," the binary translation of the branch qualifier, and "0xb1946cdc52464a61cba42fe4e0f5232b," the global transaction ID, for the Adaptive Server transaction ID "0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0":

```
1> select xa_gtrid("0000000A_IphIT596iC7bF2#AUfkzaM_8DY6OE0", 0)
2> go

       ...

------------------------------------------------------------------------
          0xb1946cdc52464a61cba42fe4e0f5232b

(1 row affected)
```

Usage

If an external transaction is blocked on Adaptive Server and you are using sp_lock and sp_transactions to identify the blocking transaction, you can use the XA transaction manager to terminate the global transaction. However, when you execute sp_transactions, the value of *xactname* it returns is in ASCII string format, while XA Server uses an undecoded binary value. Using xa_gtrid thus allows you to determine the gtrid portion of the transaction name in a format that can be understood by the XA transaction manager.

xa_gtrid returns:

- The translation version of tis string that follows the first "_" (underscore) and preceeds either the second "_" or end-of-string value, whichever comes first.
- NULL if the transaction ID cannot be decoded, or is in an unexpected format.

xa_bqual is often used with xa_gtrid.

Note xa_gtrid does not perform a validation check on the xid.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Any user can use xa_gtrid.

See also

Function xa_bqual

Stored procedures sp_lock, sp_transactions

PART 4

New features for Adaptive Server release 12.5.3a

Part 4 includes these feature descriptions:

CHAPTER 30

New Features in Adaptive Server 12.5.3a

This document describes new features available for Adaptive Server version 12.5.3a.

Column encryption

Adaptive Server authentication and access control mechanisms ensure that only properly identified and authorized users can access data. Data encryption further protects sensitive data on disk or in archives against theft or security breaches.

Data encryption in Adaptive Server allows you to encrypt data at the column level. You encrypt only the sensitive data, thereby minimizing processing overhead.

The encrypted columns feature in Adaptive Server is easier to use than encryption in the middle tier or in the client application. You use sql statements to create the encryption keys and specify columns for encryption. Adaptive Server handles key generation and storage. Encryption and decryption of data occur automatically and transparently as you write and read the data in encrypted columns. No application changes are required, and there is no need to purchase third-party software.

When data is encrypted, it is stored as ciphertext. Non-encrypted data is stored as plaintext.

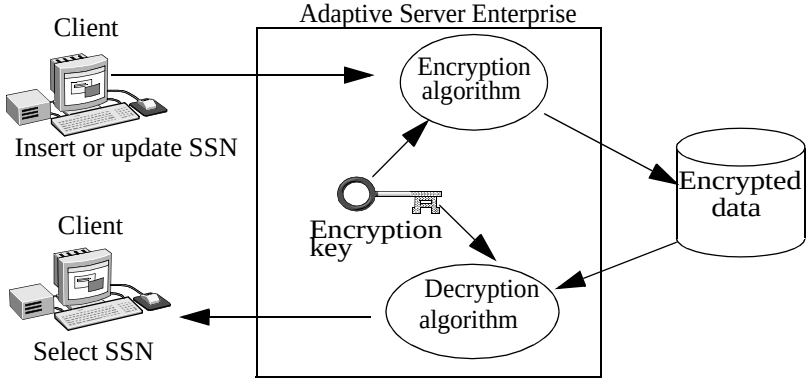
The encryption functionality is also contained in:

- Sybase Central and the Adaptive Server plug-in. See the *Sybase Central User's Manual* for more information.
- sybmigrate (the migration tool), bulk copy, and CIS, which are documented in the *Adaptive Server Enterprise System Administration Guide*.
- Replication Server. Use the *Replication Server Administration Guide* for information on encryption when replicating.

Overview

Figure 30-1 displays a high-level look at encryption and decryption processing in Adaptive Server. In this example, the Social Security Number (SSN) is being updated and encrypted:

Figure 30-1: Encryption and decryption in Adaptive Server



To create encryption keys, use create encryption key, which:

- Internally creates a symmetric key for the specified length of the key using the Security Builder Crypto™ API.
- Stores the key in encrypted form in the system catalog *sysencryptkeys*.

Adaptive Server keeps track of which key is used to encrypt a given column. Column encryption uses a symmetric encryption algorithm, which means that the same key is used for encryption and decryption.

When you insert or update data in an encrypted column, Adaptive Server transparently encrypts the data immediately before writing the row. When you select from an encrypted column, Adaptive Server decrypts the data after reading it from the row. Integer and floating point data are encrypted in a canonical form:

- Most significant bit (MSB) format for integer data.
- Institute of Electrical and Electronics Engineers (IEEE) floating point standard with MSB format for floating point data.

Data encrypted on one platform may be decrypted on another platform, provided that both platforms use the same character set.

To use encrypted columns in Adaptive Server:

1 Install the license option ASE_ENCRYPTION. See the *Adaptive Server Enterprise Installation Guide* for information.

2 Enable encryption in Adaptive Server Enterprise:

```
sp_configure 'enable encrypted columns', 0|1
```

0 – disable encryption.

1 – enable encryption.

Restart the server after you set this option.

If you turn off this option in a server that contains encrypted columns, any commands against these columns fail with an error message. Both the configuration parameter and the license option are needed to use encrypted columns. Only the System Security Officer can enable encrypted columns.

3 Set the system encryption password for a database using sp_encryption. See “Setting the system encryption password” on page 408.

4 Create the key for encrypting columns. See “Creating encryption keys” on page 408.

5 Specify the columns for encryption. See “Specifying encryption on new tables” on page 414 and “Encrypting data in existing tables” on page 415

6 Grant decrypt permission to users who must see the data. See “Permissions for decryption” on page 416.

Setting the system encryption password

The System Security Officer uses sp_encryption to set the system encryption password. The system password is specific to the database where sp_encryption is executed, and its encrypted value is stored in the sysattributes system table in that database.

```
sp_encryption system_encr_passwd, password
```

Password can be as many as 64 bytes in length, and is used by Adaptive Server to encrypt all keys in that database. Once the system encryption password has been set, you need not specify this password to access keys or data.

The system encryption password must be set in every database where encryption keys are created.

The System Security Officer can change the system password by using sp_encryption and supplying the old password.

```
sp_encryption system_encr_passwd, password [ , old_password]
```

When the system password is changed, Adaptive Server automatically reencrypts all keys in the database with the new password.

Creating and managing encryption keys

Adaptive Server generates the encryption keys and stores them in the database in encrypted form. Key owners grant table owners permission to encrypt columns in the current database with a named key.

Creating encryption keys

All the information related to keys and encryption is encapsulated by the create encryption key, which allows you to specify the encryption algorithm and key size, the key's default property, as well as the use of an initialization vector or padding during the encryption process.

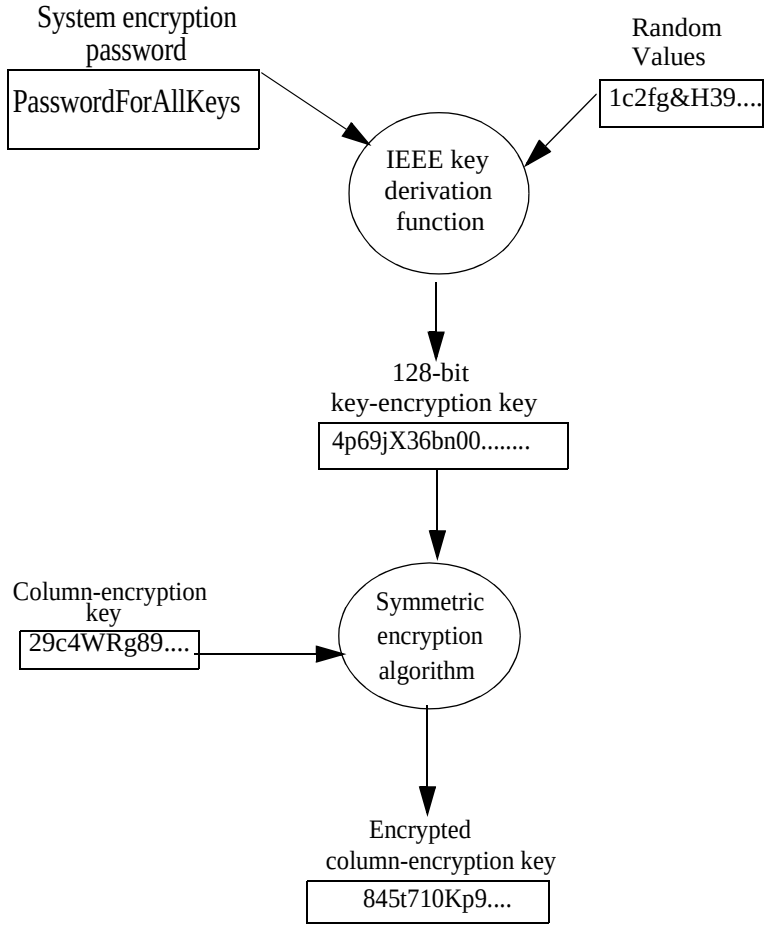
Column encryption in Adaptive Server uses the Advanced Encryption Standard (AES) symmetric key encryption algorithm, with available key sizes of 128, 192, and 256 bits. Random key generation and cryptographic functionality is provided by Security Builder Crypto™ API.

You can create separate keys for each encrypted column. Keys can be shared between columns, but each column can have only one key. To encrypt one column using an initialization vector and one column without using an initialization vector, create two separate keys, one that specifies use of an initialization vector and one that specifies no initialization vector.

The System Security Officer uses the as default clause in the create encryption key to set a default encryption key for the database. The default key is used whenever the encrypt qualifier is used without a key name on create table or alter table.

To securely protect key values, Adaptive Server uses the system encryption password to generate a 128-bit key-encrypting key, which in turn is used to encrypt the newly created key. The column-encryption key is stored in encrypted form in the *sysencryptkeys* system table.

Figure 30-2: Encrypting user keys



The syntax for create encryption key is:

```
create encryption key keyname [as default] for algorithm
[with [keylength num_bits]
[init_vector [null | random]]
[pad [null | random]]]
```

where:

- *keyname* – must be unique in the user's table, view, and procedure name space in the current database.

- as default – allows the System Security Officer to create a database default key for encryption. This enables the table creator to specify encryption without using a keyname on create table, alter table and select into. Adaptive Server uses the default key from the same database. The default key may be changed. See "alter encryption key" on page 440.
- *algorithm* – Advanced Encryption Standard (AES) is the only algorithm supported. AES supports key sizes of 128 bits, 192 bits, and 256 bits and a block size of 16 bytes.
- keylength *num_bits* – the size, in bits, of the key to be created. For AES, valid key lengths are 128, 192, and 256 bits. The default keylength is 128 bits.
- init_vector random – specifies use of an initialization vector during encryption. When an initialization vector is used by the encryption algorithm, the ciphertext of two identical pieces of plaintext are different, which prevents a cryptanalyst from detecting patterns of data. Use of an initialization vector can add to the security of your data.

  An initialization vector has some performance implications. Index creation, and optimized joins and searches, can be performed only on a column whose encryption key does not specify an initialization vector. See "Performance considerations" on page 423.
- init_vector null – omits the use of an initialization vector when encrypting. This makes the column suitable for supporting an index.

  The default is to use an initialization vector, that is, init_vector random. Use of an initialization vector implies using a cipher block chaining (CBC) mode of encryption; setting init_vector null implies the electronic code book (ECB) mode.
- pad null – is the default. It omits random padding of data.

  You cannot use padding if the column must support an index.
- pad random – data is automatically padded with random bytes before encryption. You can use padding instead of an initialization vector to randomize the ciphertext. Padding is suitable only for columns whose plaintext length is less than half the block length. For the AES algorithm the block length is 16 bytes.

For example, to specify a 256-bit key called "safe_key" as the database default key, the System Security Officer enters:

```
create encryption key safe_key as default for AES with
        keylength 256
```

The following example creates a 128-bit key called “salary_key” for encrypting columns using random padding:

```
create encryption key salary_key for AES with
        init_vector null pad random
```

This example creates a 192-bit key named “mykey” for encrypting columns using an initialization vector:

```
create encryption key mykey for AES with keylength 192
        init_vector random
```

The System Security Officer has default permission to create encryption keys and may grant that permission to other users.

For example:

```
grant create encryption key to key_admin_role
```

Using encryption keys

When you specify a column for encryption, you may use a named key from the same database, or from a different database. If you do not specify a named key, the column is automatically encrypted with the default key from the same database.

Encrypting with a key from a different database provides a distinct security advantage because it protects against access to both keys and encrypted data in the event of theft of a database dump. To access data, access to both the database archive containing data and the database archive containing encryption keys is necessary. Administrators can also protect database dumps with different passwords, making unauthorized access even more difficult.

Encrypting with a key from a different database needs special care to avoid data and key integrity problems in distributed systems. Carefully coordinate database dumps and loads. If you use a named key from a different database, Sybase recommends that:

- When you dump the database containing encrypted columns, you also dump the database where the key was created. This is necessary if new keys have been added since the last dump.
- When you dump the database containing an encryption key, dump all databases containing columns encrypted with that key. This keeps the encrypted data in sync with the available keys.

The System Security Officer can identify all the columns encrypted with a given key using sp_encryption. See “sp_encryption” on page 441.

Granting permissions on keys

The key owner must grant select permission on the key before another user can specify the key in a create table, alter table, and select into statements. For the database default key, the owner is the System Security Officer. Grant select permission on keys only on an "as needed" basis.

The following example allows users with db_admin_role to use the encryption key "safe_key" when specifying encryption on create table and alter table statements:

```
grant select on safe_key to db_admin_role
```

Users who process encrypted columns through insert, update, delete, and select do not need select permission on the encryption key.

Changing the key

As part of your information security policy, periodically change the keys used to encrypt columns. Create a new key using create encryption key, then use alter table...modify to encrypt the column with the new key

In the following example, assume that the creditcard column is already encrypted. The alter table command decrypts and reencrypts the creditcard value for every row of customer using cc_key_new.

```
create encryption key cc_key_new for AES

alter table customer modify creditcard encrypt with
cc_key_new
```

See "alter table" on page 442.

Encrypting data

You can encrypt these datatypes:

- int, smallint, tinyint
- float4 and float8
- decimal and numeric
- char and varchar
- binary and varbinary

The underlying type of encrypted data on disk is varbinary. See "Length of encrypted columns" on page 419 for more information about the length of the varbinary data.

Null values are not encrypted.

Specifying encryption on new tables

To encrypt columns in a new table, using this column option on the create table statement:

[encrypt [with [*database*.[*owner*].]*keyname*]]

keyname – identifies a key created using create encryption key. The creator of the table must have select permission on *keyname*. If *keyname* is not supplied, Adaptive Server looks for a default key created using the as default clause on the create encryption key. See "create table" on page 445 for the complete syntax for create table.

The following example creates two keys: a database default key, which uses default values for init_vector, pad and keylength, and a named key, cc_key, with non-default values. The ssn column in the employee table is encrypted using the default key, and the creditcard column in the customer table is encrypted with cc_key:

```
create encryption key new_key as default for AES
create encryption key cc_key for AES with
          keylength 256
          init_vector null
          pad random

create table employee_table (ssn char(15) encrypt)

create table customer (creditcard char(20)
          encrypt with cc_key)
```

Creating indexes on encrypted columns

You can create an index on an encrypted column if the encryption key has been specified with no initialization vector or random padding. An error occurs if you execute create index on an encrypted column that has an initialization vector or random padding. Indexes on encrypted columns are useful for equality and non-equality matches but not for range searches or ordering.

In the following example, cc_key specifies encryption without using an initialization vector or padding. This allows an index to be built on any column encrypted with cc_key:

```
create encryption key cc_key for AES
           with init_vector null

create table customer(custid int,
           creditcard varchar(16) encrypt with cc_key)

create index cust_idx on customer(creditcard)
```

Encrypting data in existing tables

To encrypt columns in existing tables use this column option on the alter table statement:

[encrypt [with [*database*.[*owner*].]*keyname*

keyname – identifies a key created using create encryption key. The creator of the table must have select permission on *keyname*. If *keyname* is not supplied, Adaptive Server looks for a default key created using the as default clause on the create encryption key. See the *Adaptive Server Enterprise Reference Manual* for the complete alter table syntax.

Note Encrypting a column in an existing table on which a trigger has been created causes the alter table to fail with an error. You must drop the trigger before you alter the table for encryption. Re-create the trigger after the alter table .. encrypt.

You can alter the encryption property on a column at the same time you alter other attributes such as datatype and nullability. You can also add an encrypted column using alter table.

For example:

```
alter table customer modify custid encrypt with cc_key
alter table customer add address varchar(50) encrypt
     with cc_key
```

See “alter table” on page 444 for the complete syntax.

Decrypting data

Permissions for decryption

You must have these two permissions to select cleartext data from an encrypted column or to search or join on an encrypted column:

- select permission on the column
- decrypt permission on the column used in the target list and in where, having, order by, update, and other such clauses

The table owner uses grant decrypt to grant explicit permission to decrypt one or more columns in a table to other users, groups and roles. Decrypt permission may be implicitly granted when a procedure or view owner grants:

- exec permission on a stored procedure that selects from an encrypted column where the owner of the procedure also owns the table containing the encrypted column.
- decrypt permission on a view column that selects from an encrypted column where the owner of the view also owns the table.

In both cases, decrypt permission need not be granted on the encrypted column in the base table.

The syntax is:

grant decrypt on [*owner*.] *table*[(*column[{* ,*column*}])] to *user* | *group* | *role*

Granting decrypt permission at the table level grants decrypt permission on all encrypted columns in the table.

Grant decrypt permission on all encrypted columns in the customer table:

```
grant decrypt on customer to accounts_role
```

The following example shows the implicit decrypt permission of user2 on the ssn column of the base table “employee”. user1 sets up the employee table and the employee_view as follows:

```
create table employee (ssn varchar(12)encrypt,
        dept_id int, start_date date, salary money)

create view emp_salary as select
            ssn, salary from employee

grant select, decrypt on emp_salary to user2
```

user2 has access to decrypted Social Security Numbers when selecting from the emp_salary view:

```
select * from emp_salary
```

Revoking decryption permission

You can revoke a user's decryption permission using:

revoke decrypt on [*owner*.] *table*[(*column*[{,*column*}])] from *user* | *group* | *role*

For example:

```
revoke decrypt on customer from public
```

Dropping encryption and keys

Dropping encryption

If you are a table owner, you can drop the encryption on a column by using alter table with the decrypt option.

The syntax is:

drop encrypton key [*database*.[*owner*].]*keyname*

For example, drop encryption on the creditcard column in the customer table:

```
alter table customer modify creditcard decrypt
```

Dropping keys

The System Security Officer and key owners can drop keys. A key can be dropped only if there are no encrypted columns in any database that use the key. You cannot check suspect and offline databases for columns encrypted by the key. The command issues a warning message naming the unavailable database but the command does not fail. When the database is brought online, any tables whose columns were encrypted with the dropped key will not be usable. To restore the key, the System Administrator must load a dump of the dropped key's database from a time that precedes when the key was dropped.

Drop an encryption key using:

drop encryption key [*database*.[*owner*].]*keyname*

For example:

```
drop encryption key cust.dbo.cc_key
```

select into command

By default, select into creates a target table without encryption even if the source table has one or more encrypted columns. The select into requires column-level permissions, including decrypt, on the source table.

Encrypt columns on the new table by using:

```
select [all|distinct] < column_list>
        into table_name
        [(colname encrypt [with [[database.[owner].]keyname]
                [, colname encrypt
                [with [[ database.]owner].]keyname]])]
         from table_name | view_name
```

You can encrypt a specific column in the target table even if the data was not encrypted in the source table. If the column in the source table is encrypted with the same key specified for the target column, Adaptive Server bypasses the decryption step on the source table and the encryption step on the target table.

The rules for encryption on a target table are the same as those for the encrypt specifier in the create table on the source table in regard to:

- Allowable datatypes on the columns to be encrypted
- The use of the database default key when the keyname is omitted
- The requirement for select permission on the key used to encrypt the target columns.

For example, encrypt the creditcard column:

```
select creditcard, custid, sum(amount) into
     #bigspenders
     (creditcard encrypt with cust.dbo.new_cc_key)
     from daily_xacts group by creditcard
     having sum(amount) > $5000
```

Length of encrypted columns

During create table and alter table operations, Adaptive Server calculates the maximum internal length of the encrypted column. To make decisions on schema arrangements and page sizes, the Database Owner must know the maximum length of the encrypted columns.

AES is a block cipher algorithm. The length of encrypted data for block cipher algorithms is a multiple of the block size of the encryption algorithm. For AES, the block size is 128 bits, or 16 bytes. Therefore, encrypted columns occupy a minimum of 16 bytes with additional space for:

- The initialization vector. If used, the initialization vector adds 16 bytes to each encrypted column. By default, the encryption process uses an initialization vector. Specify init_vector null on create encryption key to omit the initialization vector.
- The length of the plaintext data. If the column type is char, varchar, binary, or varbinary, the data is prefixed with 2 bytes before encryption. No extra space is used by the encrypted column unless the additional 2 bytes result in the ciphertext occupying an extra block.
- A sentinel byte, which is a byte appended to the ciphertext to safeguard against the database system trimming trailing zeros.

Table 30-1: Ciphertext lengths

User-specified column type	Input data length	Init vector?	Internal column type	Encrypted data length
tinyint, smallint, or int	1, 2, or 4	No	varbinary(17)	17
tinyint, smallint, or int	1, 2, or 4	Yes	varbinary(33)	33
tinyint, smallint, or int	0 (null)	No	varbinary(17)	0
float, float(4), real	4	No	varbinary(17)	17
float, float(4), real	4	Yes	varbinary(33)	33
float, float(4), real	0 (null)	No	varbinary(17)	0
float(8), double	8	No	varbinary(17)	17
float(8), double	8	Yes	varbinary(33)	33
float(8), double	0 (null)	No	varbinary(17)	0
numeric(10,2)	3	No	varbinary(17)	17
numeric (10,2)	3	Yes	varbinary(33)	33
numeric (38,2)	18	No	varbinary(33)	33
numeric (38,2)	18	Yes	varbinary(49)	49
numeric (38,2)	0 (null)	No	varbinary(33)	0
char, varchar (100)	1	No	varbinary(113)	17
char, varchar(100)	14	No	varbinary(113)	17
char, varchar(100)	15	No	varbinary(113)	33
char, varchar(100)	15	Yes	varbinary(117)	49
char, varchar(100)	31	Yes	varbinary(117)	65
char, varchar(100)	0 (null)	Yes	varbinary(117)	0
binary, varbinary(100)	1	No	varbinary(113)	17
binary, varbinary(100)	14	No	varbinary(113)	17
binary, varbinary(100)	15	No	varbinary(113)	33
binary, varbinary(100)	15	Yes	varbinary(117)	49
binary, varbinary(100)	31	Yes	varbinary(117)	65
binary, varbinary(100)	0 (null)	Yes	varbinary(117)	0

char and binary are treated as variable-length datatypes and are stripped of blanks and zero padding before encryption. Any blank or zero padding is applied when the data is decrypted.

Note The column length on disk increases for encrypted columns, but the increases are invisible. For example, sp_help shows only the original size.

Auditing encrypted columns

You can audit DDL commands that relate to encrypted columns, such as creating or dropping an encryption key. Also, when you create a table the audit record contains the name of the encrypted column and the corresponding encryption key. A database-wide audit option enables you to group and manage the audit records of encrypted columns and keys.

Auditing options

The following table is excerpted from Adaptive Server System Administration Guide and shows the new commands audited with existing event options and the new event options.

Table 30-2: Auditing options, requirements, and examples

Options	*login_name*	*object_name*	Database to be in to set the option	Command or access being audited
encryption_key (database-specific)	all	Database to be audited	Any	alter encryption key create encryption key drop encryption key create table drop table alter table sp_encryption
	Example `sp_audit "encryption_key", "all", "pubs2", "on"` (Audits all the above commands in the pubs2 database.)			

Audit values

Table 30-3 lists the values that appear in the event column, arranged by sp_audit option. The “Information in extrainfo” column describes information that might appear in the extrainfo column of an audit table, based on the categories described in Table 30-3.

Table 30-3: Values in event and extrainfo columns

Audit option	Command to be audited	event	Information in extrainfo output

alter	alter table	3	*Keywords or options:*
			`ADD/DROP/MODIFY COLUMNS`
			`REPLACE COLUMN`
			`ADD CONSTRAINT`
			`DROP CONSTRAINT`
			If 1 or more encrypted columns are added, *keywords* will contain:
			`ADD/DROP/MODIFY COLUMNS column1/keyname1, [,column2/keyname2]`
			where *keyname* is the fully qualified name of the key.

create	create table	10	For encrypted columns, keywords contain column names and keynames. `EK column1/keyname1[,column2 keyname2]` where EK is a prefix indicating that subsequent information refers to encryption keys and *keyname* is the fully qualified name of the key.
encryption_key	sp_encryption	106	*Keywords* contain `ENCR_ADMIN system_encr_passwd password` ******** if password is set the first time, and contains `ENCR_ADMIN system_encr_passwd password` ******** ******** if the password is subsequently changed
	create encryption key	107	*Keywords contain:* algorithm Name-bitlength/IV [RANDOM|NULL]/PAD [RANDOM|NULL] For example: `AES-128/IV RANDOM/PAD NULL`

	alter encryption key	108	*Keywords contain*:
			`NOT DEFAULT`
			if key no longer the default key.
			`DEFAULT`
			if the key is made the default key

	drop encryption key	109	

New event names and numbers

You can query the audit trail for specific audit events. Use audit_event_name with the *event id* as a parameter.

audit_event_name(*event_id*)

In Table 30-4 the new event numbers and names are listed.

Table 30-4: New event numbers

Event number	Event names output
106	`Encrypted Column Administration`
107	`Create Encryption Key`
108	`Alter Encryption Key`
109	`Drop Encryption Key`

Performance considerations

Encryption is a CPU-intensive operation that may introduce a performance overhead to your application in terms of CPU usage and the elapsed time of commands that use encrypted columns. The overhead depends on the number of CPUs and Adaptive Server engines, the load on the system, the number of concurrent sessions accessing the encrypted data, and the number of encrypted columns referenced in the query. The encryption key size and the length of the encrypted data are also factors. In general, the larger the key size and the wider the data, the higher the CPU usage in the encryption operation.

This section discusses the performance implications of searching encrypted columns, and how Adaptive Server Enterprise optimizes processing of encrypted data to minimize the number of encryption and decryption operations.

Indexes on encrypted columns

You can create an index on an encrypted column, provided that the column's encryption key does not specify the use of an initialization vector or random padding. Using an initialization vector or random padding results in identical data encrypting to different patterns of ciphertext, which prevents the index from enforcing uniqueness.

Indexes on encrypted data are useful for equality and non-equality matching of data but not for data ordering, range searches, or finding minimum and maximum values. If Adaptive Server is performing an order-dependent search on an encrypted column, it cannot execute an indexed lookup on encrypted data. Instead, the encrypted column in each row must be decrypted and then searched. This process slows the processing of data.

Joins on encrypted columns

Adaptive Server optimizes the joining of two encrypted columns by performing ciphertext comparisons if the following conditions apply:

- The joining columns have the same datatype. char and varchar are considered the same datatypes, as are binary and varbinary.
- For int and float types, the columns have the same length. For numeric and decimal types, the columns have the same precision and scale.
- The joining columns are encrypted with the same key.
- The joining columns are not part of an expression. For example, a ciphertext join cannot be performed on the join where t.encr_col1 = s.encr_col1 +1.
- The encryption key was created with init_vector and pad set to NULL.
- The join operator is '=' or '<>'.
- The data has the default sort order.

For example, this sets a schema to join on ciphertext:

```
create encryption key new_cc_key for AES
        with init_vector NULL
create table customer
        (custid int,
        creditcard char(16) encrypt with new cc_key)
create table daily_xacts
       (cust_id int, creditcard char(16) encrypt with
        new_cc_key, amount money........)
```

You can also set up indexes on the joining columns:

```
create index cust_cc on customer(creditcard)

create index daily_cc on daily_xacts(creditcard)
```

Adaptive Server executes the following select statement to total a customer's daily charges on a given credit card without decrypting the creditcard column in either the customer or the daily_xacts table.

```
select sum(d.amount) from daily_xacts d, customer c
     where d.creditcard = c.creditcard and
     c.custid = 17936
```

Constant valued search arguments and encrypted columns

For equality and non-equality comparison of an encrypted column to a constant value, Adaptive Server optimizes the column scan by encrypting the constant value once, rather than decrypting the encrypted column in each row of the table. The same restrictions listed in “Joins on encrypted columns” on page 424 apply.

Adaptive Server cannot make use of an index to perform a range search on an encrypted column; it must decrypt each row before performing data comparisons. If a query contains other predicates, Adaptive Server selects the most efficient join order which often leaves searches against encrypted columns last, on the smallest data set.

If your query has more than one range search where there is no useful index, write the query so that the range search against the encrypted column is last. For example, the following query searches for Social Security Numbers of taxpayers in Rhode Island with incomes above $100, 000. The range search of the zipcode column is positioned before the range search of the encrypted adjusted gross income column:

```
select ss_num from taxpayers
       where zipcode like '02%' and
       agi_enc > 100000
```

Movement of encrypted data as ciphertext

As much as possible, Adaptive Server optimizes the copying of encrypted data by copying ciphertext instead of decrypting and reencrypting the data. This applies to select into, bulk copy, and replication.

System tables

syscolumns

In the syscolumns system table, these columns describe encryption properties:

Field	Type	Values	Description
encrtype	int	null	Type of data in encrypted form.
encrlen	int	null	Length of encrypted data.
encrkeyid	int	null	Object id of key.
encrkeydb	varchar (30)	null	Database name where the encryption key resides. NULL if key resides in the same database as the encrypted column.
encrdate	datetime	null	Creation date, copied from sysobjects.crdate.

sysobjects

sysobjects has an entry for each key with type EK (encryption key).

For cross-database key references, syscolumns.encrdate matches sysobjects.crdate.

encrkeyid in *sysencryptkeys* matches the id column in *sysobjects*.

sysencryptkeys

Each key created in a database, including the default key, has an entry in the database-specific system catalog sysencryptkeys.

Table 30-5: sysencryptkeys

Field	Type	Description
id	int	Encryption key ID
ekalgorithm	int	Encryption algorithm
type	smallint	Identifies the key type. The values are EK_SYMMETRIC and EK_DEFAULT.
status	int	Internal status information
eklen	smallint	User-specified length of key
value	varbinary(1282)	Encrypted value of a key. contains a symmetric encryption of the key. To encrypt keys, Adaptive Server uses AES with a 128-bit key from the system encryption password.
uid	int null	Not used
eksalt	varbinary(20)	Contains random salt used to validate decryption of the encryption key.
ekpairid	int null	Not used
pwdate	datetime null	Not used
expdate	int null	Not used
ekpwdwarn	int null	Not used

ddlgen utility changes

ddlgen supports generation of DDL statements for encrypted keys. To specify a key, use:

<dbName>.<owner>.<keyName>

The new type EK, for encryption key, is for generating the DDL to create an encryption key and to grant permissions on it. ddlgen generates encrypted column information and a grant decrypt statement, with the DDL of a table.

This example generates DDL for all encrypted keys in a database “accounts” on a machine named “HARBOR” using port 1955:

```
ddlgen -Uroy -Proy123 -SHARBOR:1955 -TEK
-Naccounts.dbo.%
```

Alternatively, you can specify the database name with the -D option:

```
    ddlgen -Uroy -Proy134 -SHARBOR:1955 -TEK -Ndbo.%
      -Daccounts
-----------------------------------------------------
----------------------
```

```
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------
-----------------------
      print 'ssn_key'

create encryption key accounts.dbo.ssn_key
      for AES
      with keylength 128
      init vector random
go


------------------------------------------------------
-----------------------
-- DDL for EncryptedKey 'ek1'
------------------------------------------------------
-----------------------
print 'ek1'

create encryption key accounts.dbo.ek1 as default
     for AES
     with keylength 192
     init vector NULL
go

use accounts
go

grant select on accounts.dbo.ek1 to acctmgr_role
go
```

ddlgen also has an extended option to generate the create encryption key that specifies the key's encrypted value as represented in *sysencryptkeys*. The option is -XOD and can be used if you must synchronize encryption keys across servers for data movement. For example, to make cc_key on server "PACIFIC" available on server "ATLANTIC", execute ddlgen using -XOD on "PACIFIC" as follows:

```
ddlgen -Sfred -Pget2work -SPACIFIC:8532 -TEK -Nsales.dbo.cc_key -XOD
```

ddlgen output is:

```
------------------------------------------------------------------------
-- DDL for EncryptedKey 'cc_key'
------------------------------------------------------------------------
print 'cc_key'

create encryption key sales.dbo.cc_key
```

```
      for AES
with keylength 128
passwd 0x0000E1D8235FEBEB118901
init_vector NULL
keyvalue 0xF772B99CE547D2932A12E0A83F2114848BD93F38016C068D720DDEBAC4DF8AA001
keystatus 32
go
```

Next change the key generated by create encryption key to specify the target database on "ATLANTIC", and run the command on the target server. cc_key is now available on server "ATLANTIC" to decrypt data that is moved from "PACIFIC" to "ATLANTIC".

See *Adaptive Server Enterprise Utility Guide* for more information about ddlgen syntax options, and see *Replication Server Administration Guide* for examples of using ddlgen with replicated databases.

Replicating encrypted data

If your site replicates schema changes, the following DDL statements are replicated:

- alter encryption key
- create table and alter table with extensions for encryption
- create encryption key
- grant and revoke create encryption key
- grant and revoke select on the key
- grant and revoke decrypt on the column
- sp_encryption system_encr_passwd
- drop encryption key

The keys are replicated in encrypted form.

If your system does not replicate DDL, you must manually synchronize encryption keys at the replicate site. ddlgen supports a special form of create encryption key for replicating the key's value.

insert and update replicate encrypted columns in encrypted form, which safeguards replicated data while Replication Server processes it in stable queues on disk.

See the *Replication Server Administration Guide* for information on encryption when replicating.

Bulk copy (*bcp*)

bcp transfers encrypted data in and out of databases in either plaintext or ciphertext form. By default, bcp copies plaintext data. bcp processes plaintext data files as follows:

- Data is automatically encrypted by Adaptive Server before insertion when executing bcp in. Slow bcp is used. The user must have insert and select permission on all columns.
- Data is automatically decrypted by Adaptive Server when executing bcp out. select permission is required on all columns; in addition, decrypt permission is required on the encrypted columns.

This example copies the "customer" table out as plaintext data in native machine format:

```
bcp uksales.dbo.customer out uk_customers -n -Uroy
-Proy123
```

Use the -C option for bcp to copy the data as ciphertext. When copying ciphertext, you may copy data across different operating systems. If you are copying character data as ciphertext, both platforms must support the same character set.

The -C option for bcp allows administrators to run bcp when they lack decrypt permission on the data. When the -C option is used, bcp processes data as follows:

- Data is assumed to be in ciphertext format during execution of bcp in, and Adaptive Server performs no encryption. Use the -C option with bcp in only if the file being copied into Adaptive Server was created using the -C option on bcp out. The ciphertext must have been copied from a column with exactly the same column attributes and encrypted by the same key as the column into which the data is being copied. Fast bcp is used. The user must have insert and select permission.
- Data is copied out of Adaptive Server without decryption on bcp out. The ciphertext data is in hexadecimal format. The user must have select permission on all columns. For copying ciphertext, decrypt is not required on the encrypted columns.

- Encrypted char or varchar data retains the character set used by Adaptive Server at the time of encryption. If the data is copied in ciphertext format to another server, the character set used on the target server must match that of the encrypted data copied from the source. The character set associated with the data on the source server when it was encrypted is not stored with the encrypted data and is not known or converted on the target server.

  The System Administrator must verify that the character sets match on both the source and the target servers. You can also perform the bcp without the -C option and the character set issue is not encountered.

  The -J option for character set conversion can not be used with the -C option.

The following example copies the "customer" table. The cc_card column is copied out as ciphertext. Other columns are copied in character format. User "roy" is not required to have decrypt permission on customer cc_card.

```
 bcp uksales.dbo.customer out uk_customers -C -c -Uroy
-Proy123
```

Warning! You may use the -C flag with bcp out on a view only if the view qualification does not search encrypted columns.

Component Integration Services (CIS)

By default, encryption and decryption is handled by the remote Adaptive Server. CIS makes a one-time check for encrypted columns on the remote Adaptive Server. If the remote Adaptive Server supports encryption, CIS updates the local *syscolumns* catalog with the encrypted-column-related metadata.

- create proxy_table automatically updates *syscolumns* with any encrypted-column information from the remote tables.
- create existing table automatically updates *syscolumns* with any encrypted-column metadata from the remote tables. The encrypt keyword is not allowed in the *columnlist* for create existing table. CIS automatically marks columns as encrypted if it finds any encrypted columns on the remote table.
- create table at location with encrypted columns is not allowed.

- alter table is not allowed on encrypted columns for proxy tables.
- select into existing brings the plaintext from the source and inserts it into destination table. The local Adaptive Server then encrypts the plaintext before insertion into any encrypted columns.

The following columns are updated from the remote server's *syscolumns* catalog:

- encrtype – type of data on disk.
- encrlen – length of encrypted data.
- status2 – status bits that indicate that column is encrypted.

load and *dump* databases

dump and load work on the ciphertext of encrypted columns. This behavior ensures that the data for encrypted columns remains encrypted while on disk. dump and load pertain to the whole database. Default keys and keys created in the same database are dumped and loaded along with the data to which they pertain.

If the loading database contains encryption keys used in other databases, load does not succeed unless the new syntax with override is used.

```
load database key_db from "/tmp/key_db.dat" with override
```

If your keys are in a separate database from the columns they encrypt, Sybase recommends that:

- When you dump the database containing encrypted columns, you also dump the database where the key was created. This is necessary if new keys have been added since the last dump.
- When you dump the database containing an encryption key, dump all databases containing columns encrypted with that key. This keeps the encrypted data in sync with the available keys.
- After loading the database containing the encryption keys and the database containing the encrypted columns, bring both databases on line at the same time.

If you load the database containing the keys into a different-named database, errors will result when you access the encrypted columns in other databases. To change the database name of the keys' database, take the following steps:

- Before dumping the database containing the encrypted columns, use alter table to decrypt the data.
- Dump the databases containing keys and encrypted columns.
- After loading the databases, use alter table to re-encrypt the data with the keys in the newly-named database.

The consistency issues between encryption keys and encrypted columns are similar to those for cross-database referential integrity. See "Cross-database constraints and loading databases" in the *Adaptive Server Enterprise System Administration Guide*.

Do not attempt to load any dumps containing encrypted data into prior versions of Adaptive Server. Load the database into an Adaptive Server version 12.5.3a and remove any encryption from it. Perform the dump and then load the database into an Adaptive Server with a prior version. See “Downgrade procedure” on page 436.

See “Creating and managing encryption keys” on page 408 for more information on keys.

unmount database

When columns are encrypted by keys from other databases, unmount all related databases as a set. The interdependency of the databases containing the encrypted columns and the databases containing the keys is similar to the interdependency of databases that use referential integrity.

You use the override option to unmount a database containing columns encrypted by a key in another database.

With the following commands the encryption key created in key_db has been used to encrypt columns in col_db. These commands successfully unmount the named databases:

```
unmount database key_db, col_db
unmount database key_db with override
unmount database col_db with override
```

These commands will fail with an error message without the override:

```
unmount database key_db

unmount database col_db
```

quiesce database

You can use quiesce database when the database contains the encryption key.

You must use with override to quiesce a database whose columns are encrypted with keys used in other databases.

`quiesce database` ***`key_db, col_db`*** is allowed where *key_db* is the database with the encryption key and *col_db* is the database with a table that has a column encrypted with the key in *key_db*.

For example, the following commands will succeed where key_db contains the encryption key used to encrypt columns in col_db:

```
quiesce database key_tag hold key_db for external
         dump to "/tmp/keydb.dat"

quiesce database encr_tag hold col_db for external dump
      to "/tmp/col.dat" with overridewith override

quiesce database col_tag hold key_db, col_db for
    external dump to "/tmp/col.dat"
```

Drop database

To prevent accidental loss of keys, Adaptive Server fails drop database if it contains keys currently used to encrypt columns in other databases. Before dropping the database containing the encryption keys you must first remove the encryption or drop the database containing the encrypted columns.

In the following example key_db is the database where the encryption key resides and col_db is the database containing the encrypted columns:

```
drop database key_db, col_db
```

Adaptive Server raises an error and fails to drop key_db. The drop of col_db succeeds. To drop both databases, drop col_db first:

```
drop database col_db, key_db
```

sybmigrate

sybmigrate is the migration tool used to migrate data from one server to another.

By default, sybmigrate migrates encrypted columns in ciphertext format. This avoids the overhead of decrypting data at the source and encrypting at the target. In some cases, sybmigrate chooses the reencrypt method of migration, decrypting data at the source and encrypting at the target.

For databases with encrypted columns, sybmigrate:

1 Migrates the system encryption password. If you specify not to migrate the system encryption password, sybmigrate migrates the encrypted columns using the reencrypt method instead of migrating the ciphertext directly.

2 Migrates the encryption keys. You may select the keys to migrate. sybmigrate automatically selects keys in the current database used to encrypt columns in the same database. If you have selected migration of the system encryption password, sybmigrate migrates the encryption keys using their actual values. The key values from the *sysencryptkeys* system table have been encrypted using the system encryption password and these are the values that are migrated. If you have not migrated the system encryption password, sybmigrate migrates the keys by name, to avoid migrating keys that will not decrypt correctly at the target. Migrating the key by name causes the key at the target to be created with a different key value from the key at the source.

3 Migrates the data. By default, the data is transferred in its ciphertext form. Ciphertext data can be migrated to a different operating system. Character data requires that the target server uses the same character set as the source.

sybmigrate works on a database as a unit of work. If your database on the source server has data encrypted by a key in another database, migrate the key's database first.

sybmigrate chooses to reencrypt migrated data when:

- Any keys in the current database are specifically not selected for migration, or already exist in the target server. There is no guarantee that the keys at the target are identical to the keys are the source, so the migrating data must be reencrypted.
- The system password was not selected for migration. When the system password at the target differs from that at the source, the keys cannot be migrated by value. In turn, the data cannot be migrated as ciphertext.
- The user uses the following flag:

```
sybmigrate -T 'ALWAYS_REENCRYPT'
```

Reencrypting data can slow performance. A message to this effect is written to the migration log file when you perform migration with reencryption mode.

To migrate encrypted columns, you must have both sa_role and sso_role enabled.

Downgrade procedure

If you have never configured enable encrypted columns in your server, you need not take any action before using an older version of Adaptive Server with 12.5.3a databases. One way to verify that you have never configured encrypted columns is to check that the system table *sysencryptkeys* does not exist in any database.

All databases should be backed up prior to the downgrade procedure.

Before downgrading a server that has been configured for encrypted columns, you must either drop or modify any tables with encrypted columns to remove encryption. You then run sp_encryption remove_catalog, which verifies that there are no encrypted columns in each database and then removes the system table *sysencryptkeys*. The new columns in *syscolumns* added for 12.5.3a are ignored by an older binary and need not be removed

To downgrade from a 12.5.3a server to an earlier version of 12.5.x:

1 If encrypted columns are not currently enabled, the System Security Officer executes:

```
sp_configure 'enable encrypted columns',1
```

2 Use drop or alter to decrypt all tables with encrypted columns in all databases. The System Security Officer runs the following command in each database where encryption keys were created to list all encryption keys created in that database:

```
sp_encryption help
```

For each key listed, the System Security Officer runs the following to see a list of columns encrypted with a particular key:

sp_encryption help, <*keyname*>, 'display_cols'

For each encrypted column, one of the following steps must be performed:

- alter table to decrypt the encrypted columns
- alter table to drop the encrypted columns
- drop the table containing the encrypted column

- drop the encryption key

3 To guarantee that no other user can access Adaptive Server while a system table is removed, restart the server in single-user mode. See the *Adaptive Server Enterprise Utility Guide.*

4 A user with sso_role and sa_role must execute the following system stored procedure, which removes the *sysencryptkeys* catalog from each database:

```
sp_encryption remove_catalog
```

If a database is unavailable, the command prints an error and exits. If columns encrypted by any key in *sysencryptkeys* exist, the command does not drop *sysencryptkeys*, but prints an error or warning and continues with the next database.

If sp_encryption is successful in removing *sysencryptkeys*, it also removes these rows from *sysattributes* in each database:

- The record of the upgrade item that added *sysencryptkeys*
- The system encryption password for the database

5 Drop the system stored procedure sp_encryption from the *sybsystemprocs* database.

6 Shut down the server. You can now use a 12.5.x Adaptive Server binary from a pre-12.5.3a version.

To reenable encrypted columns, when rolling forward from a downgraded 12.5.3a server back to 12.5.3a, configure enable encrypted columns. Upon restarting the 12.5.3a server, the *sysencryptkeys* system table is installed in each database.

Replication issues with downgrade

When downgrading a server that has replication enabled on databases that contain encrypted data, you must do one of the following before you start the downgrade procedure:

1 Ensure that all replicated data in the primary database transaction log has been successfully transferred to the standby or replicate database. The process for doing this is application dependent.

2 Truncate the transaction log in the primary database, and zero the RS locator for that database in the Replication Server. Use the following commands:

In the primary database run:

```
sp_stop_rep_agent primary_dbname
        dbcc settrunc ('ltm', 'ignore')
        dump tran  primary_dbname with truncate_only
        dbcc settruc ('ltm', 'valid')
```

Shutdown Replication Server. In the RSSD for the Replication Server run:

```
rs_zeroltm primary_servername, primary_dbname
```

New commands

create encryption key

All the information related to keys and encryption is encapsulated by create encryption key, which allows you to specify the encryption algorithm and key size, the key's default property, as well as the use of an initialization vector or padding during the encryption process

Adaptive Server uses Security Builder Crypto™ for key generation and encryption.

The System Security Officer has default permission to create encryption keys and may grant that permission to other users.

Syntax

create encryption key [[*database.[owner].]keyname* [as default] for *algorithm*
[with [keylength *num_bits*]
[init_vector [NULL | random]]
[pad [NULL | random]]]

- *keyname* – must be unique in the user's table, view, and procedure name space in the current database.
- as default – allows the System Security Officer to create a database default key for encryption. This enables the table creator to specify encryption without using a keyname on create table, alter table and select into. Adaptive Server uses the default key from the same database. The default key may be changed. See "alter encryption key" on page 440.
- *algorithm* – Advanced Encryption Standard (AES) is the only algorithm supported. AES supports key sizes of 128 bits, 192 bits, and 256 bits and a block size of 16 bytes.
- keylength *num_bits* – the size, in bits, of the key to be created. For AES, valid key lengths are 128, 192, and 256 bits. The default keylength is 128 bits.

- init_vector random – specifies use of an initialization vector during encryption. When an initialization vector is used by the encryption algorithm, the ciphertext of two identical pieces of plaintext are different, which prevents a cryptanalyst from detecting patterns of data. Use of an initialization vector can add to the security of your data.

  An initialization vector has some performance implications. Index creation, and optimized joins and searches, can be performed only on a column whose encryption key does not specify an initialization vector. See "Performance considerations" on page 423.

- init_vector null – omits the use of an initialization vector when encrypting. This makes the column suitable for supporting an index.

  The default is to use an initialization vector, that is, init_vector random. Use of an initialization vector implies using a cipher block chaining (CBC) mode of encryption; setting init_vector null implies the electronic code book (ECB) mode.

- pad null – is the default. It omits random padding of data.

  You cannot use padding if the column must support an index.

- pad random – data is automatically padded with random bytes before encryption. You can use padding instead of an initialization vector to randomize the ciphertext. Padding is suitable only for columns whose plaintext length is less than half the block length. For the AES algorithm the block length is 16 bytes.

For example, to specify a 256-bit key called "safe_key" as the database default key, the System Security Officer enters:

```
create encryption key safe_key as default for AES with
       keylength 256
```

The following example creates a 128-bit key called "salary_key" for encrypting columns using random padding:

```
create encryption key salary_key for AES with
        init_vector null pad random
```

This example creates a 192-bit key named "mykey" for encrypting columns using an initialization vector:

```
create encryption key mykey for AES with keylength 192
       init_vector random
```

alter encryption key

To change the default encryption key, enter:

alter encryption key *key1* as default

If a default key already exists it no longer has the default property. *key1* becomes the default key.

If *key1* is the default key, you can remove the default designation for *key1* as follows:

alter encryption key *key1* as not default

If *key1* is not the default key, the command returns an error.

alter encryption key as default or not default can be executed only by the System Security Officer and cannot be granted to other users

drop encryption key

The key owner and the System Security Officer can drop encryption keys. The command fails if any column in any database is encrypted using the key.

Syntax

drop encryption key [*database*.[*owner*].]*keyname*

grant create encryption key

The System Security Officer grants permission to create encryption keys.

Syntax

grant create encryption key to *user* | *role*| *group*

revoke create encryption key

The System Security Officer can revoke permission from other users, groups, and roles to create encryption keys.

Syntax

revoke create encryption key from *user* | *role* | *group*

grant decrypt

The table owner or the System Security Officer grants decrypt permission on a table or a list of columns in a table.

Syntax

```
grant decrypt on [ owner. ]tablename[(columnname [{,columname}])]
    to user | group | role
```

Note grant all on a table or column does not grant decrypt permission.

revoke decrypt

The table owner or the System Security Officer revokes decrypt permission on a table or a list of columns in a table.

Syntax

```
revoke decrypt on [owner.] tablename[(columnname [{,columname}])]
from user | group | role
```

sp_encryption

The System Security Officer sets the system encryption password using sp_encryption. The system password is specific to the database where sp_encryption is executed, and its encrypted value is stored in the sysattributes system table in that database.

```
sp_encryption system_encr_passwd, 'password'
```

The password specified using sp_encryption can be up to 64 bytes in length, and is used by Adaptive Server to encrypt all keys in that database. You need not specify this password to access keys or data.

The system encryption password must be set in every database where encryption keys are created.

The System Security Officer can change the system password by using sp_encryption and supplying the old password.

```
sp_encryption system_encr_passwd, 'password' [ , 'old_password']
```

When the system password is changed, Adaptive Server automatically reencrypts all keys in the database with the new password.

sp_encryption help

sp_encryption help displays the key's name, owner, size and encryption algorithm. It also indicates whether the key has been designated as the database default key, and whether encryption with this key uses random padding or an initialization vector.

```
sp_encryption help [, keyname [, display_cols]]
```

When sp_encryption help is run by a user with sso_role, the key properties of all keys in the database are displayed. When run by a user without sso_role, the key properties are displayed for only those keys for which the user has select permission in that database.

sp_encryption help, *keyname* displays the properties of keyname. If the command is run by a user without sso_role, the user must have select permission on the key.

sp_encryption help, keyname, display_cols may be run only by a user with sso_role. It lists the columns encrypted by keyname.

Changes to command syntax

The addition of the encrypted columns feature has changed or added syntax to the commands in this section.

alter table

Use alter table to encrypt or decrypt existing data or to add an encrypted column to a table.

Syntax

Encrypt a column:

```
alter table tablename add column_name
        encrypt [with [database.[owner].]keyname
```

Decrypt an existing column:

```
[decrypt [with [database.[owner].]keyname]]
```

keyname – identifies a key created using create encryption key. The creator of the table must have select permission on *keyname*. If *keyname* is not supplied, Adaptive Server looks for a default key created using create encryption key or alter encryption key as default.

Example

Create an encryption key and encrypt ssn column in existing “employee” table.

```
alter table employee modify ssn
         encrypt with ssn_key
grant decrypt on employee(ssn) to hr_manager_role,
         hr_director_role
```

Use alter table to change an encryption key. When the encrypt qualifier is used on a column that is already encrypted, Adaptive Server decrypts the column and re-encrypts it with the new key. This operation may take a significant amount of time if the table contains a large number of rows.

create table

Use the encrypt qualifier to set up encryption on a table column.

Syntax

```
create table tablename (colname datatype [default_clause]
        [encrypt [with [database.[owner].]keyname]])
```

keyname – identifies a key created using create encryption key. The creator of the table must have select permission on *keyname*. If *keyname* is not supplied, Adaptive Server looks for a default key created using the as default clause on create encryption key or alter encryption key.

Example

Create an employee table with encryption:

```
create table employee_table (ssn char(15) null encrypt)
```

enable encrypted columns configuration parameter

The configuration parameter enable encrypted columns must be set to 1 to use the encryption functionality. If the configuration option is turned off in a server that contains encrypted columns, any commands against those columns fail with an error. Both the configuration parameter and the license option are needed to enable encryption.

```
sp_configure 'enable encrypted columns', 1
```

load database

If the loading database contains encryption keys used in other databases, load does not succeed unless with override is used.

```
load database key_db from "/tmp/key_db.dat" with override
```

select into

select into requires column level permissions, including decrypt, on the source table.

Syntax

Encrypt columns on the new table using this syntax:

```
select [all|distinct] < column_list>
        into target_table
        [(colname encrypt [with [database.[owner].]keyname]
        [,colname encrypt
        [with [database.[owner].]keyname]])]
    from tabname | viewname
```

Example

Encrypting creditcard column in the table.

```
select creditcard, custid, sum(amount) into
       #bigspenders
       (creditcard encrypt with
          cust.database.new_cc_key)
       from daily_xacts group by creditcard
       having sum(amount) > $5000
```

dbcc

dbcc checkcatalog includes the following additional consistency checks:

1 For each encryption key row in *sysobjects*, *sysencryptkeys* is checked for the existence of a row defining that key.

2 For each column in *syscolumns* marked for encryption, the existence of the key is checked in *sysobjects* and *sysencryptkeys*.

Full syntax for commands

The following information shows the full syntax for the commands covered in this bulletin.

alter encryption key

alter encryption key *key1* as default | not default

alter table

alter table [[*database*.[*owner*].*table_name*
{ add *column_name datatype*
[default {*constant_expression* | user | null}]
{identity | null | not null}
[off row | in row]
[[constraint *constraint_name*]
{ { unique | primary key }
[clustered | nonclustered]
[asc | desc]
[with { fillfactor = *pct*,
max_rows_per_page = *num_rows*,
reservepagegap = *num_pages* }]
[on *segment_name*]
| references [[*database*.]*owner*.]*ref_table*
[(*ref_column*)]
[match full]
| check (*search_condition*)] ... }

```
        [encrypt [with [database .] owner ] .]keyname
        [, next_column]...
    | add {[constraint constraint_name]
    {  unique | primary key}
            [clustered | nonclustered]
            (column_name [asc | desc]
        [, column_name [asc | desc]...])
        [with { fillfactor = pct,
                max_rows_per_page = num_rows,
                reservepagegap = num_pages}]
        [on segment_name]
    | foreign key (column_name [{, column_name}...])
        references [[database.]owner.]ref_table
        [(ref_column [{, ref_column}...])]
        [match full]
    | check (search_condition)}
    | drop {column_name [, column_name]...
        | constraint constraint_name }
    | modify column_name datatype [null | not null]
            [encrypt | [with [database .]owner].] keyname
    | modify column_name datatype [null | not null
            [encrypt | [with [database .]owner].] keyname]
                |decrypt
            [, next_column]...
    | replace column_name
        default { constant_expression | user | null}
        | partition number_of_partitions
    | unpartition| { enable | disable } trigger
    | lock {allpages | datarows | datapages } }
    | with exp_row_size=num_bytes
    | [ alter_partition_clause ]
    [ partition_clause ]
```

create table

```
create table [database .[owner ].]table_name (column_name datatype)
    [default {constant_expression  | user | null}]
    {[{identity | null | not null}]
        [off row | [ in row [ (size_in_bytes) ] ]
    [[constraint constraint_name ]
        {{unique | primary key}
        [clustered | nonclustered] [asc | desc]
        [with { fillfactor = pct,
                max_rows_per_page = num_rows, }
                reservepagegap = num_pages }]
        [on segment_name]
        | references [[database .]owner .]ref_table
            [(ref_column )]
            [match full]
            | check (search_condition)}]}
```

```
        [match full]...
    [encrypt [with [[ database.] owner] . ] keyname]
    | [constraint constraint_name]
        {{unique | primary key}
            [clustered | nonclustered]
            (column_name [asc | desc]
                [{, column_name [asc | desc]}...])
            [with { fillfactor = pct
                max_rows_per_page = num_rows ,
                reservepagegap = num_pages } ]
            [on segment_name]
        |foreign key (column_name [{,column_name}...])
            references [[database.]owner.]ref_table
                [(ref_column [{, ref_column}...])]
                [match full]
        | check (search_condition) ... }
    [{, {next_column | next_constraint}}...])
    [lock {datarows | datapages | allpages }]
    [with { max_rows_per_page = num_rows,
            exp_row_size = num_bytes,
            reservepagegap = num_pages,
            identity_gap = value } ]
    [ table_lob_clause ]
    [ on segment_name ]
    [ [ external table ] at pathname ]
    [ partition_clause ]
```

select

```
into_clause ::=
    into [[database.]owner.]table_name
    [(colname encrypt [with [database.] owner].] keyname
            [, colname encrypt [with [database. ]owner] . ]
                keyname])]
    [ lock {datarows | datapages | allpages } ]
    [ with into_option [, into_option] ...]

    into_option ::=
        | max_rows_per_page = num_rows
        | exp_row_size = num_bytes
        | reservepagegap = num_pages
        | identity_gap = gap
        [existing table table_name]
        [[external type] at "path_name"
        [column delimiter delimiter]]
```

Internet protocol version 6

Use trace flag 7841, which allows Adaptive Server to determine IPv6 availability, and make Adaptive Server IPv6-aware.

IPv6 is available on.

- Sun Solaris 32- and 64-bit platforms
- Windows
- HP 32- and 64-bit platforms

For more information about how to set up and administer IPv6-enabled networking on your platform, see your operating system documentation.

Real Time Messaging Services

Real Time Messaging Services for Adaptive Server version 12.5.3a provides support for both TIBCO JMS and IBM WebSphere MQSeries.

PAM support in 64-bit Adaptive Server on AIX

Adaptive Server version 12.5.3a on AIX 64-bit supports Pluggable Authentication Module-based User Authentication (PAMUA). Even though 12.5.3a 64-bit Adaptive Server on AIX 5.1, IBM also supports PAM on 64-bit applications on AIX 5.2 only. Sybase recommends that you upgrade to AIX 5.2 to use this feature. Contact your operating system representive to make sure you have the latest patch for PAM available on your IBM host.

64-bit PAM libraries shipped with AIX 5.1 automatically load 64bit libraries from `/usr/lib/security/64`. To enable PAMUA in 64-bit Adaptive Server 12.5.3a on AIX5.1, supply the full path name of the PAM module in */etc/pam.conf file*. The following example illustrates how to specify the module pam_aix on an AIX 5.1 machine.

Adaptive Server authorization required `/usr/lib/security/64/pam_aix`

Feature matrix

Figure 30-3 shows the various features available in Adaptive Server version 12.5.3a for different platforms.

Figure 30-3: Feature and platform compatibility matrix

Operating System	Sol32	Sol64	HP32	HP64	AIX64	Linux x86	Windows x86
Options							
Encrypted Columns	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a
High Availability	*	*	*	*	*	*	*
Distributed Transaction	*	*	*	*	*	*	*
XML Management	*	*	*	*	*	*	*
Java Option	*	*	*	*	*	*	*
Native XML	*	*	*	*	*	*	*
Java Based XML	*	*	*	*	*	*	*
Web Services	*	*	*	*	*	*	*
Security & Dir Services	*	*	*	*	*	*	*
LDAP Server Directory	*	*	*	*	new for 12.5.3a	*	*
LDAP User Authentication	*	*	*	*	new for 12.5.3a	*	*
Secure Socket Layer	*	*	*	*	*	*	*
Cybersafe Kerberos	*	*					*
MIT Kerberos	*	*		new for 12.5.3a	new for 12.5.3a	*	
Platform Native Kerberos	*	*					
Fine Grained Access Control	*	*	*	*	*	*	*
Pluggable Authentication Modu	*	*			new for 12.5.3a	*	
Content Management	*	*	*	*	*	*	*
Enhanced Full Text Search	*	*	*	*	*	*	*
Real Time Messaging	*	*		*	*	*	*
JMS support	*	*		*	*	*	*
Websphere MQ support	new for 12.5.3a	new for 12.5.3a		new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	
Disaster Recovery	*		*			*	*
Features Included with ASE							
IPv6	*	*	new for 12.5.3a	new for 12.5.3a			new for 12.5.3a
Cross Platform Dump and Loa	*	*	*	*	*	*	*
Job Scheduler	*	*	*	*	*	*	*
ASE Replicator	*	*	*	*	*	*	*

- Indicates supported feature. A blank cell indicates that feature is not available on that platform.
- High Availability support indicates Active-Active support.

  Active-Passive support for High Availability is limited to Solaris/SPARC platform only.

- Option availability varies on various editions. Please check the Adaptive Server Enterprise Data Sheet for more information about the differences between various editions.

PART 5

New features for Adaptive Server release 12.5.3

Part 5 includes these feature descriptions:

CHAPTER 31

New Language Support

Adaptive Server version 12.5.3 supports new languages.

EFTS supported languages

The Enhanced Full-Text Search Specialty Data Store (EFTS) now includes language support for:

- Traditional Chinese on the Windows and Solaris platforms
- Arabic, Hebrew, Thai, and Russian on the Linux platform

CHAPTER 32 *top n* functionality

Adaptive Server version 12.5.3 introduces support for top n functionality.

top n

Use top n with select...into statements to limit the number of rows inserted in the target table. This is different from set rowcount, which is ignored during a select...into.

top n functionality works with Microsoft SQL Server, Adaptive Server Anywhere, and Adaptive Server IQ.

Use the top n clause in a select command to limit the number of rows in the result set to the number of rows specified by "n," which is an unsigned 32-bit value between 0 through 2^{32}-1 (4GB-1 or 4,294,967,295). Zero indicates "no" rows.

Adaptive Server Enterprise version 12.5.3 supports the top n clause in outer query select statements, but not in the select list of a subquery. This differs from Microsoft SQL Server. Any attempt to use the top n clause with Adaptive Server in a subquery yields a syntax error.

select statement

This is the syntax for the top n clause in a select statement:

```
select
all_distinct_clause
top unsigned_integer
select_list
into_clause
from_where_clause
group_by_clause
having_clause
```

This example selects the first 5 rows from table t1:

```
select top 5 col1 from t1
```

union statements

The syntax to use the top n clause in a select statement with a union is:

```
select top 2 column_name from table_name
union all
select top 3 column_name from table_name
```

This returns five rows, assuming t1 has at least 2 rows and t2 has at least 3 rows. The top limit applies to the individual selects that form a union, not to the union as a whole.

update statement

In an update statement, insert the top n clause immediately after the keyword:

```
UPDATE
TOP unsigned_integer
object_identifier
SET
set_clause_list
from_where_clause
for_clause
abstract_plan_clause
```

delete statement

In a delete statement, insert the top n clause immediately after the keyword:

```
DELETE
TOP unsigned_integer
result_table
from_where_clause
for_clause
abstract_plan_clause
```

Usage

- When you use top n with delete, update, or in a view, you cannot specify ordering. If there is an implied order on the table from a clustered index, that order applies, otherwise, the results are unpredictable and they can be in any order.
- When used with cursors, top n limits the overall size of the result set. If you specify set cursor rowcount, top n limits the results of a single fetch.
- When a view definition contains select top n and a query with a where clause uses it, the results may be inconsistent.

CHAPTER 33

Secure Sockets Layer

Adaptive Server version 12.5.3 supports the following Secure Socket Layer (SSL) features:

Overview

Adaptive Server Enterprise version 12.5.3 includes these changes for Secure Sockets Layer (SSL):

- New cipher suites using the Advanced Encryption Standard (AES) algorithm
- New options to sp_ssladmin to set preferences for cipher suites accepted by Adaptive Server
- New global variable *@@ssl_ciphersuite* to tell the client which cipher suite was chosen by the SSL handshake

These enhancements let the system security officer manage SSL more efficiently, and allow client applications to determine the encryption algorithms used on their connection.

Advanced Encryption Standard (AES) algorithm

Two new cipher suites using AES are available in Adaptive Server version 12.5.3:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA

AES is a FIPS-197 approved standard for symmetric encryption. The AES algorithm provides the strongest encryption available for Adaptive Server.

The AES algorithm and dependent cipher suites used in SSL are available by default in Adaptive Server 12.5.3 and Open Client 12.5.1. No action is required by users or administrators to use this algorithm unless algorithms are currently restricted client applications. If you have clients that currently restrict algorithms, you may want to consider using AES algorithm instead.

Setting SSL cipher suite preferences

In Adaptive Server version 12.5.3, sp_ssladmin has two new command options, lsciphers and setciphers. With these new options, the set of cipher suites that Adaptive Server uses can be restricted, giving control to the system security officer over the encryption algorithms that may be used by client connections to the server or outbound connections from Adaptive Server. The default behavior for use of SSL cipher suites in Adaptive Server is the same as in earlier versions; it uses an internally defined set of preferences for cipher suites.

To display the values for any set cipher suite preferences, enter:

```
sp_ssladmin lsciphers
```

To set a specific cipher suite preference, enter:

```
sp_ssladmin setciphers, {"FIPS" | "Strong" | "Weak" |
"All" | quoted_list_of_ciphersuites }
```

where:

- "FIPS" – is the set of encryptions, hash, and key exchange algorithms that are FIPS-compliant. The algorithms included in this list are AES, 3DES, DES, and SHA1.
- "Strong" – is the set of encryption algorithms using keys longer than 64 bits.
- "Weak" – is the set of encryption algorithms from the set of all supported cipher suites that are not included in the strong set.
- "All" – is the set of default cipher suites.

- quoted_list_of_ciphersuites – specifies a set of cipher suites as a comma-separated list, ordered by preference. Use double quotes to mark the beginning and end of the list. The quoted list can include any of the predefined sets as well as individual cipher suite names. Unknown cipher suite names cause an error to be reported, and no changes are made to preferences.

The detailed contents of the predefined sets are in Table 33-1.

sp_ssladmin setciphers sets cipher suite preferences to the given ordered list. This restricts the available SSL cipher suites to the specified set of "FIPS", "Strong", "Weak", "All", or a quoted list of cipher suites. This takes effect on the next listener started, and requires that you restart Adaptive Server to ensure that all listeners use the new settings.

You can display any cipher suite preferences that have been set using sp_ssladmin lsciphers. If no preferences have been set, sp_ssladmin lsciphers returns 0 rows to indicate no preferences are set and Adaptive Server uses its default (internal) preferences.

Table 33-1: Predefined cipher suite sets in Adaptive Server 12.5.3

Set name	Cipher suite names included in the set
FIPS	TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_DES_CBC_SHA TLS_DHE_DSS_WITH_DES_CBC_SHA TLS_DHE_RSA_WITH_DES_CBC_SHA TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
Strong	TLS_RSA_WITH_AES_256_CBC_SHA TLS_RSA_WITH_AES_128_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_RC4_128_SHA TLS_RSA_WITH_RC4_128_MD5 TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA TLS_DHE_DSS_WITH_RC4_128_SHATLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA

Set name	Cipher suite names included in the set
Weak	TLS_RSA_WITH_DES_CBC_SHA
	TLS_DHE_DSS_WITH_DES_CBC_SHA
	TLS_DHE_RSA_WITH_DES_CBC_SHA
	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
	TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA
	TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA
	TLS_RSA_EXPORT_WITH_RC4_40_MD5
	TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
	TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
	TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
All	TLS_RSA_WITH_AES_256_CBC_SHA
	TLS_RSA_WITH_AES_128_CBC_SHA
	TLS_RSA_WITH_3DES_EDE_CBC_SHA
	TLS_RSA_WITH_RC4_128_SHA
	TLS_RSA_WITH_RC4_128_MD5
	TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
	TLS_DHE_DSS_WITH_RC4_128_SHA
	TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
	TLS_RSA_WITH_DES_CBC_SHA
	TLS_DHE_DSS_WITH_DES_CBC_SHA
	TLS_DHE_RSA_WITH_DES_CBC_SHA
	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
	TLS_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA
	TLS_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA TLS_RSA_EXPORT_WITH_RC4_40_MD5
	TLS_RSA_EXPORT_WITH_DES40_CBC_SHA
	TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
	TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA

Warning! Sybase recommends that you do not use any cipher suites that are not included in predefined sets because they pose potential security vulnerabilities.

Table 33-2: Cipher suites to avoid

Reason to avoid	Cipher suites
Cipher suites with "anon" for certificate exchange algorithm; the server is not authenticated by its certificate.	TLS_DH_anon_WITH_3DES_EDE_CBC_SHA TLS_DH_anon_WITH_RC4_128_MD5 TLS_DH_anon_WITH_DES_CBC_SHA TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA TLS_DH_anon_EXPORT_WITH_RC4_40_MD5
Cipher suites with "NULL" for the symmetric key encryption algorithm do not encrypt data for transmission on the network.	TLS_RSA_WITH_NULL_SHA TLS_RSA_WITH_NULL_MD5

Examples

On initial start-up, before any cipher suite preferences have been set, no preferences are shown by sp_ssladmin lscipher.

```
1> sp_ssladmin lscipher
2> go
```

Output:

```
 Cipher Suite Name   Preference
-----------------   ----------
(0 rows affected)
(return status = 0)
```

The following example specifies the set of cipher suites that use FIPS algorithms.

```
sp_ssladmin setcipher, 'FIPS'
```

A preference of 0 (zero) sp_ssladmin output indicates a cipher suite is not used by Adaptive Server. The other, nonzero numbers, indicate the preference order in which Adaptive Server uses the algorithm during the SSL handshake. The client side of the SSL handshake chooses one of these cipher suites that matches its list of accepted cipher suites.

This example uses a quoted list of cipher suites to set preferences in Adaptive Server:

```
1> sp_ssladmin setcipher,
'TLS_RSA_WITH_AES_128_CBC_SHA,
```

```
TLS_RSA_WITH_AES_256_CBC_SHA'
2> go
```

Other considerations

When you upgrade to Adaptive Server version 12.5.3, the cipher suite preferences are the server defaults, and sp_ssladmin option lscipher displays no preferences. The server uses its default preferences, those defined by "All". The system security officer should consider the security policies employed at the site and the available SSL cipher suites to decide whether to restrict cipher suites and which cipher suites are appropriate for the security policies.

If you downgrade from Adaptive Server version 12.5.3 to an earlier version, any cipher suite preferences are ignored, and the Adaptive Server defaults for that version are used.

If you have set SSL cipher suite preferences and want to remove all preferences from the server and use default preferences, delete the preferences from their storage location in system catalogs using the following commands:

```
1> sp_configure 'allow updates to system tables', 1
2> go

1>  delete from master..sysattributes where class=24
2>  go

1> sp_configure 'allow updates to system tables', 0
2> go
```

These commands can be executed only by the system security officer or system administrator.

@@ssl_ciphersuite

The Transact-SQL global variable *@@ssl_ciphersuite* enables users to see which cipher suite was chosen by the SSL handshake and to verify that an SSL or a non-SSL connection was established.

For example, an isql connection using SSL protocol displays the cipher suite chosen for it.

```
1>  select @@ssl_ciphersuite
```

```
2> go
```

Output:

```
------------------------------
TLS_RSA_WITH_AES_128_CBC_SHA

(1 row affected)
```

SSL on Linux 32-bit

SSL is now fully supported on Linux 32-bit platform, including the use of SSL with CIS RPCs.

CHAPTER 34

Dumping and Loading Databases Across Platforms

Adaptive Server version 12.5.3 introduces cross platform dump and load capabilities for platforms with different endian architecture.

Overview

Adaptive Server Enterprise version 12.5.2 supported the dump and load of databases across platforms with the same endian architecture.

With Adaptive Server version 12.5.3, you can now dump and load databases across platforms with different endian architecture. This means you can perform dump database and load database from either a big endian platform to a little endian platform, or from a little endian platform to a big endian platform.

In a big-endian system, the most significant byte of storage, such as integer or long, has the lower address. The reverse is true for a little-endian system.

There is no syntax change with dump or load database in version 12.5.3. Adaptive Server automatically detects the architecture type of the originating system of the database dump file during a load database, then performs the necessary conversions. Loads from older versions, such as 11.9 and 12.0, are also supported. The dump and load can be from 32-bit to 64-bit platforms, and vice versa.

Platforms supported:

Big-endian	Sun Solaris	IBM AIX	Silicon Graphics	HP-UX on HPPA, HPIA	MAC OS X
Little-endian	Linux IA	Windows	HP True 64*	Sun Solaris x86	

Note On True64, the XPDL feature does not work as the backupserver dump is incompatible with other platforms.

Dump and load across platforms with the same endian architecture

When you perform dump database and load database across platforms with the same endian architecture, user and system data do not require conversions. There are no limitations on operations with the dump and load of a database.

Stored procedures and other compiled objects are recompiled from the SQL text in *syscomments* at the first execution after the load database for certain combination platforms.

Dump and load across platforms with different endian architecture

Adaptive Server allows a dump and load database between big endian and little endian architectures, and vice versa.

Dumping a database

1 Execute dbcc checkdb and dbcc checkalloc to verify the database runs cleanly.

2 To prevent concurrent updates from open transactions by other processes during dump database, use sp_dboption to place the database in a single-user mode.

3 Flush statistics to systabstats using sp_flushstats.

4 Wait for 10 to 30 seconds, depending on the database size and activity.

5 Run checkpoint against the database to flush updated pages.

6 Run dump database.

Loading a database

Once you load the database, Adaptive Server automatically identifies the endian type on the dump file and performs all necessary conversions during load database and online database.

Note After Adaptive Server converts the index rows, the order of index rows may be incorrect. Adaptive Server marks following indexes on user tables as suspect indexes during online database.

- Nonclustered index on an all-pages locked table
- Clustered index on data-only locked table
- Nonclustered index on data-only locked table

See sp_post_xpload for information about checking and rebuilding indexes on user tables.

Restrictions

- dump transaction and load transaction is not allowed across platforms.
- dump database and load database to or from a remote backupserver are not supported across platforms.
- You cannot load a password-protected dump file across platforms.
- If you perform dump database and load database for a parsed XML object, you must parse the text again after the load database is completed.
- You cannot perform dump database and load database across platforms on Adaptive Server versions earlier than 11.9.

- Adaptive Server cannot translate embedded data structures stored as binary, varbinary, or image columns.
- load database is not allowed on the master database across platforms.
- Stored procedures and other compiled objects are recompiled from the SQL text in syscomments at the first execution after the load database.

  If you do not have permission to recompile from text, the person who does must recompile from text using dbcc upgrade_object to upgrade objects.

Note If you migrate login records in *syslogins* system table in the master database from Solaris to Linux, you can use bcp with character format. The login password from the Solaris platform is compatible on Linux without a trace flag. Beginning with 15.0.2, any password that uses SHA-256 can be moved between platforms using bcp. sp_displaylogin is used to determine the password encryption algorithm used by a particular login.

Performance considerations

Due to the design of indexes within a data server, which provides an optimum search path, index rows are ordered for fast access to the table's data row. Index rows which contain row identifiers (RIDs), are treated as binary to achieve a fast access to the user table.

Within the same architecture platform, the order of index rows remains valid and search order for a selection criteria takes its normal path. However, when index rows are translated across different architectures, the order which optimization was done is invalidated. This results in an invalid index on user tables when a cross-platform dump and load feature is executed.

To fix indexes on the target system after load from a different architecture dump, you can use one of two methods:

1 Drop and re-create all of the indexes.

2 Use sp_post_xpload.

Since the data point and information varies from usage on indexes, the schema, user data, number of indexes, index-key length, and number of index rows, in general, it requires planning to re-create indexes on large tables, as it can be a lengthy process. sp_post_xpload validates indexes, drops invalid indexes, and re-creates dropped indexes in a single command.

Since sp_post_xpload performs many operations, it may take longer than dropping and recreating indexes. Sybase recommends that you use drop and re-create indexes on databases larger than 10G.

CHAPTER 35

Changes to the HP-UX Platform

This chapter contains information that is applicable to HP-UX for Adaptive Server Enterprise version 12.5.3.

Newly supported features on HP-UX platforms

Adaptive Server version 12.5.3 supports the following features on HP-UX platforms:

- Feedback Optimization (FBO) – Adaptive Server supports the FBO server on HP-11.23 Itanium, which improves performance.
- High Availability – on the Itanium platform, the high availability option is available on HP-UX 11.23 with HP ServiceGuard A.11.15. See *Using Sybase Failover in A High Availability System*.
- XML – services are available for the HP-UX (on HPPA platform) 32-bit and 64-bit platforms. The XML Management option must be licensed separately.

CHAPTER 36

Resource Governor

Adaptive Server version 12.5.3 provides resource limits which can help System Administrators prevent queries and transactions from monopolizing server resources.

Setting resource limits

A resource limit is a set of parameters that prevents an individual login or application from:

- Exceeding a given estimated or actual I/O costs
- Returning excessive rows on a per query basis
- Exceeding a given elapsed time on a query batch or transaction basis
- Utilizing excessive tempdb space per session

When a system administrator modifies a resource limit, all users logged in the session see the change, including the system administrator.

CHAPTER 37

Changes to Configuration Parameters for Real Time Data Services

In Adaptive Server version 12.5.3, use sp_configure to set the number of native threads and the wait time for messaging.

Setting the number of native threads

Use max native threads per engine to define the maximum number of native threads per engine the server spawns. When the limit for the native threads is reached, Adaptive Server sessions that require a native thread sleep until another session releases a native thread.

max native threads per engine

Summary information	
Default value	50
Maximum values	1000
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

Setting the wait time for messaging

Use rtm thread idle wait period to define the length of time a native thread used by Adaptive Server waits when it has no work to do. When the time set for a native thread is reached, the thread automatically fades out.

rtm thread idle wait period

Summary information	
Default value	50 seconds
Maximum value	4026531839 seconds
Status	Dynamic
Display level	Intermediate
Required role	System Administrator

CHAPTER 38

Migration Tool

Adaptive Server version 12.5.3 supports sybmigrate functionality.

Topic	Page
sybmigrate enhancements in Adaptive Server 12.5.3	477

sybmigrate enhancements in Adaptive Server 12.5.3

Note sybmigrate allows you to migrate across versions of Adaptive Server and supports source servers from 12.0 through 15.0.2.

Adaptive Server versions 12.5.3 and later allow you to specify the size and location of a work database on your target server. When migrating a database or server from a source server with Adaptive Server Enterprise versions 12.0 and later but earlier than 12.5.0.1, you must specify the size and location of a work database on the target server.

CHAPTER 39

Page Allocation Changes

Adaptive Server version 12.5.3 increases the efficiency of its page allocation procedure.

Allocating pages in Adaptive Server version 12.5.3

In versions of Adaptive Server earlier than 12.5.3, inserts into a partitioned data-only locked table with a placement index may have caused wasted space. This problem was more pronounced in tables with a higher number of partitions and with larger page sizes.

Adaptive Server version 12.5.3 avoids wasting extra space by filling up existing allocated extents in the target allocation page, even though these extents are assigned to other partitions. The net effect is that extents are allocated only when there are no free extents in the target allocation page.

If you do not want to use this space allocation mechanism, activate command line trace flag 646 (-T646).

CHAPTER 40

Importing Statistics For Proxy Tables

Adaptive Server version 12.5.3 has improved its ability to update and import statistics for remote server proxy tables.

Importing statistics

In Adaptive Server version 12.5.3, when you perform update statistics on a remote server proxy table, if the relevant table and index statistics are available, the table catalogs are imported to the local systabstats and sysstatistics.

By default, update statistics for proxy tables always attempts to import the required statistics data. However, if the statistics data is unavailable or incomplete on the remote table, Component Integration Services (CIS) reverts to the prior mechanism of gathering statistic data.

You can also force CIS to revert to the prior mechanism of gathering statistic data by turning on trace flag 11229. This allows you to obtain all data from the database, then calculating the statistics.

Limitations

Key limitations:

- The proxy table must be mapped to another Adaptive Server version 11.9 or later.

- Excludes proxy tables mapped to RPCs, external files, and system directories.
- If the remote servers are not Adaptive Server version 11.9 or later, or of another server class, CIS continues to obtain statistics data using prior mechanisms.

CHAPTER 41

Historical Server Changes

Adaptive Server version 12.5.3 allows you to send monitoring data from Historical Server to a database on a specific Adaptive Server.

Sending data to a database server

Select an Adaptive Server, and:

1 Create a database for the Historical Server to use for storing monitoring data. The default name of the database is *hs_monitoring*. If you do not want to use the default name, create a new database with the name you want to use, then change the database name in the hs_directload.sql script.

2 Run hs_directload.sql on the database. This installation script creates two catalog tables, sessions and views, and the stored procedure sp_hs_dboutput.

Starting Historical Server

The command line syntax is (the new syntax is in default font):

histserver -U*user name*
- P*password* -D*output dir*
-l*log file* -I*interfaces file*
[-d*delimiter*] [-OASE_name
] [-odatabase_name] [-f]
-uoutput_ASE_user name - poutput_ASE_password

where:

- -O *ASE_name* – the name of the target Adaptive Server.
- -o *database_name* – the database name to which the monitoring data is sent, if it is not *hs_monitoring*, identify the database name on this option.

- -u *output_ASE_ user name* – login name for the connection to the Adaptive Server with the output.
- -p *output_ASE_password* – password for the login name for the output Adaptive Server.
- -f – must be used when the -O option is specified if you want Historical Server to send data to files in the output directory as well as the database.

You must have access to Historical Server and update permissions to the target database.

You must specify:

- The destination for the monitoring data from Historical Server to a specific Adaptive Server and database, rather than a flat file.
- The user name and password for the output if they are not the same as those on Historical Server. If the user name and the password are not specified, the values specified in the -U and -P parameters are used.

The target Adaptive Server and database for the historical data must be available when Historical Server is started.

Viewing the data

The data from Historical Server is in the same format as defined when the view from the view table was created on the Adaptive Server. To store the data, Historical Server creates two system tables:

- Sessions table provides a record of every recording session that used the output database.
- Views table lists the views that were used by each recording session.
- In addition to one table for each view that is output to Adaptive Server, there are two system tables created in the Historical Server output database. These tables are:
  - The *sessions* table provides a record of every recording session that has used the output database.
  - The *views* table lists the views that were used by each recording session.

The table structures are similar to the structure of the output data files, where:

- first column is the monitoring session ID.
- second column is the monitored server name.
- third column is a date or timestamp.
- subsequent columns per data item, specified in the view definition.

This structure, for the date and timestamp, and the data items, is identical to the structure defined in the DDL scripts Historical Server provides when a bulk copy is executed on Historical Server data files into another Adaptive Server.

Changes to the dtdValidation option

For Adaptive Server Enterprise version 12.5.3, the dtdValidate option for Historical Server has been expanded to include:

- dtdValidate='no'

  No validation is performed whether the document has an embedded DTD or references an external DTD. If the document contains an embedded DTD, dtdValidate verifies that it is correct but does not validate.

- dtdValidate='yes'

  Validates only if there is an embedded DTD or a reference to an external DTD.

- dtdValidate='strict'

  The document must contain an embedded or reference to an external DTD, and validates against the DTD.

CHAPTER 42

User Connections

Adaptive Server version 12.5.3 include an updated error message and a correction on reserved sockets.

Number of user connections

Versions of Adaptive Server earlier than 12.5.3 reserved on-third of available sockets to provide for the Enterprise JavaBeans (EJB) server, whether or not EJB was configured. Once reserved, these sockets were unavailable for Adaptive Server.

With Adaptive Server version 12.5.3 ESD #2, no sockets are automatically reserved for EJB. However, you can enable trace flag 1642 to revert to the previous functionality, reserving one-third of the sockets for EJB. You must enable traceflag 1642 to set up the EJB server if the message, `"hbc_ninit: No sockets available for HBC"` appears in the error log. If the EJB server is not configured, you can ignore this message.

If the EJB server is enabled and host based communication (HBC) sockets are not available, the message `"hbc_ninit: No sockets available for HBC"` is reported. If traceflag 1642 is not enabled, then Adaptive Server using the 1642 traceflag. If the EJB server is not enabled, then no message is reported and Adaptive Server automatically disables the sockets reserved for EJB server.

User disconnections

Error 1608 now displays the host name and login name when a client connection is disconnected abnormally from Adaptive Server. The extended error information also includes additional diagnostics that the system administrator can use to analyze and confirm the cause of the disconnections. The error message appears as follows:

```
00:00000:00017:2004/11/08 16:01:21.03 kernel Cannot send, host process
disconnected: TONYI-XP 512 suid: 1
00:00000:00017:2004/11/08 16:01:23.10 server Error: 1608, Severity: 18, State:
4
00:00000:00017:2004/11/08 16:01:23.10 server A client process exited
abnormally, or a network error was encountered. Unless other errors occurred,
continue processing normally.
00:00000:00017:2004/11/08 16:01:23.10 kernel extended error information:
hostname: TONYI-XP login: sa
```

CHAPTER 43

Monitor Counters and *sp_sysmon*

In Adaptive Server version 12.5.3, some of the most commonly used monitoring tools have been enhanced.

Note Using sp_sysmon with the noclear option became the default behavior in Adaptive Server version 15.0.1 and later. For more information on updates to sp_sysmon, see "Changes to sp_sysmon" in Chapter 21, "New Features in Adaptive Server 15.0.1."

sp_sysmon noclear option

The noclear parameter for sp_sysmon and improvements to the monitor counter concurrency enable you to run multiple concurrent sessions of the sp_sysmon and other monitoring applications, such as Monitor Server, Historical Server or other sp_sysmon sessions.

When you include the noclear option in sp_sysmon, sp_sysmon does not zero out the monitor counters. As a result, data collected by other applications using the monitor counters at the same time are not affected by the sp_sysmon report.

When used with the noclear parameter, sp_sysmon utilizes a temporary table. If space in a user's default temporary database is limited, you may need to increase the size of the database to run sp_sysmon with noclear.

The report contents generated from sp_sysmon with or without the noclear should be the same. However, the sp_sysmon report may show a small amount of additional server activity when run with the noclear parameter.

A field called “Sample Mode” has been added to the sp_sysmon report header to indicate whether the report was generated using “No Clear” or “Reset Counters” mode. The report header includes the labels “Sampling Started at” and “Sampling Ended at” when run with the noclear parameter. It displays “Statistics Cleared at” and “Statistics Sampled at” when run without the noclear parameter.

When noclear is used with a specific sample interval, for any parameter in the sp_sysmon, the command runs in noclear mode. If noclear is not specified, sp_sysmon clears the counters.

Syntax

sp_sysmon interval [, noclear,[,section [, applmon]]]

Example

Report usage without clearing the counters:

```
sp_sysmon "00:01:00", kernel, noclear
sp_sysmon "00:01:00", noclear
```

Note You can use noclear only when you specify a sample interval in sp_sysmon. If you specify begin_sample or end_sample, you cannot use noclear.

Monitor counter concurrency

Adaptive Server version 12.5.3 tracks the number of applications using the monitor counters. Adaptive Server does not disable or stop data collection by the monitor counters as long as one or more applications are known to be using them. This allows applications such as sp_sysmon and the Monitor Server to operate concurrently.

This change does not require any modifications to applications using monitor counters. Adaptive Server tracks the number of different applications or user connections where the monitor counters have been enabled or disabled. The monitor counters are disabled only when all connections have disabled them.

The monitor counter usage count is not automatically decremented when an application that enabled the monitor counters logs out of Adaptive Server. This means that, if an application enables the monitor counters and does not disable them before logging out, the usage count continues to reflect the same number of users as it did before the application logged out. You can correct this by using new dbcc commands described in the following section.

New *dbcc* commands

The system administrator can now manually modify the monitor counter usage count. When an application enables monitor counters, and does not disable them before logging off Adaptive Server, the system administrator can use these commands to terminate monitor counter data collection:

```
dbcc monitor (increment, <group name>)
dbcc monitor (decrement, <group name>)
dbcc monitor (reset, <group name>)
```

Where <*group name*> can be one of the following:

- 'all'
- spinlock_s
- appl

increment and decrement increase and decrease usage counts for the monitor counters in the specified group by 1. reset sets the usage count for the monitor counters in the specified group to zero. This turns off collection of monitoring data for this group.

You can determine the usage count for all, which comprises some of the monitor counters, by running the *@@monitors_active* global variable.

The usage counts for the spinlock_s and appl groups are reported by the dbcc resource command.